Master's thesis
Master's Programme in Data Science

Generalizability of machine learning models in predicting patient deterioration

Niko Säkkinen

December 3, 2020

Supervisor(s): Assistant professor Arto Klami
Dr. Hanna Saarinen
Dr. Jyri Kivinen

Examiner(s): Assistant professor Arto Klami
Dr. Hanna Saarinen

University of Helsinki
Faculty of Science

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Degree programme
Faculty of Science	Master's Programme in Data Science

Tekijä — Författare — Author

Niko Säkkinen

Työn nimi — Arbetets titel — Title

Generalizability of machine learning models in predicting patient deterioration

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidantal — Number of pages
Master's thesis	December 3, 2020	72

Tiivistelmä — Referat — Abstract

Predicting patient deterioration in an Intensive Care Unit (ICU) effectively is a critical health care task serving patient health and resource allocation. At times, the task may be highly complex for a physician, yet high-stakes and time-critical decisions need to be made based on it. In this work, we investigate the ability of a set of machine learning models to algorithimically predict future occurrence of in-hospital death based on Electronic Health Record (EHR) data of ICU-patients. For one, we will assess the generalizability of the models. We do this by evaluating the models on hospitals the data of which has not been considered when training the models. For another, we consider the case in which we have access to some EHR data for the patients treated at a hospital of interest. In this setting, we assess how EHR data from other hospitals can be used in the optimal way to improve the prediction accuracy. This study is important for the deployment and integration of such predictive models in practice, e.g., for real-time algorithmic deterioration prediction for clinical decision support.

In order to address these questions, we use the eICU collaborative research database, which is a database containing EHRs of patients treated at a heterogeneous collection of hospitals in the United States. In this work, we use the patient demographics, vital signs and Glasgow coma score as the predictors. We devise and describe three computational experiments to test the generalization in different ways. The used models are the random forest, gradient boosted trees and long short-term memory network. In our first experiment concerning the generalization, we show that, with the chosen limited set of predictors, the models generalize reasonably across hospitals but that only a small data mismatch is observed. Moreover, with this setting, our second experiment shows that the model performance does not significantly improve when increasing the heterogeneity of the training set. Given these observations, our third experiment shows that while domain adaptation is useful, the used multi-source domain adaptation methods do not offer clear advantages.

ACM Computing Classification System (CCS):
Computing methodologies→Machine learning

Avainsanat — Nyckelord — Keywords

patient deterioration, mortality prediction, machine learning, generalization, transfer learning

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Contents

1. Introduction

Machine learning is a subfield of artificial intelligence, the idea of which, as often quoted to Arthur Samuel [54], is that computers can be given the ability to learn without being explicitly programmed to do so. What this means in practice is that a computer program learns to perform a task by using *observed data* and a *learning algorithm*. For example, when a program for *classification* is given annotated data, it learns via an algorithm an internal representation stating how to associate a *label* to each data instance. Such computer programs are realizations of *machine learning model*s.

After a machine learning model has been trained, it needs to be tested in order to understand how well it works in practice. This is usually done by evaluating its performance on new data. Here, the relevant term is *generalizability* which refers to the ability of a machine learning model to perform well on data instances which it has not seen before [42, 23]. In the usual situation, a model sees a data distribution during the learning phase, and the assumption is that at test time, the external data comes from the same distribution. It can, however, be that the distribution at the test time is different from the data distribution during the training. In this case, if the distributions of the labels are the same, the difference lies in the training and test *domains*. *Covariate shift* is the term used for such a situation and there is a need of *domain adaptation* methods [49]. A special case of this, relevant for this work, is when there are multiple different domains from which the data originates.

This thesis considers machine learning problems for the analysis of *Electronic Health Record*s (EHRs), which are digital collections of patient health information. If collected from multiple centers, EHRs are typical datasets which contain multiple domains. Each domain can be identified as a specific center with its own characteristic patient population and treatment policies [13]. A practical question then arises how to train and deploy a machine learning based model to a center for which there is no data, or a very limited amount, available. In this thesis, this question is rephrased as how do machine learning models for predicting patient deterioration generalize across hospitals. In addition, by patient deterioration, we will be referring to the task of *in-hospital mortality prediction* in the *Intensive Care*

Unit (ICU) setting.

Predicting patient deterioration in ICUs is an important task for many reasons. For one, it would be useful in triage and resource allocation [36]. Several traditional scores exits for assessing the *Severity Of Illness* (SOI) of a patient [38, 41, 62]. These scores produce a static estimate of SOI of a patient typically after the first 24 hours at an ICU. Instead, in this work, we are concerned with a real-time prediction of patient deterioration. That is, we aim to predict the risk of in-hospital death during a future time-window based on a history of patient *vitals*, i.e., vital signs. If accurate enough, such an estimate would add value over the traditional SOI estimates as it would enable more frequent care decisions [27].

The more specific research questions addressed in this work are discussed next. Our main hypothesis is that by training a machine learning based mortality prediction model with a heterogeneous set of centers improves the generalizability of the model. Supporting this, we will investigate whether or not domain adaptation methods are applicable to the multi-center ICU setting. In particular, we compare pooled single-source domain adaptation methods to the multi-source domain adaptation methods which, to our best knowledge, is a completely new contribution to the field. Moreover, in the recent years, there has been a surge of *deep learning* methods to various supervised learning tasks [35, 14]. Hence, we make an effort to address the question what are the benefits of deep learning for the in-hospital mortality prediction task similarly to [37, 1] but in another context.

In this thesis, we will find out that the studied models generalize reasonably across hospitals. In particular, we will show that the models performances systematically increase with more data available from the source hospitals. As for the specific research questions, we will find that the heterogeneity of the training set does not improve the generalization noticeably. We will also find that domain adaptation is useful but that there is no clear advantage of using multi-source methods. We suggest that this is related to using mainly the vitals as *features*. Finally, we show that with the relatively small amount of data used in this work, the deep learning approach does not outperform the more traditional tree-based machine learning approaches.

This thesis is organized as follows. In Section 2, we motivate this work by describing the need for generalizable models for ICUs. The following Section 3 goes over relevant literature. Technical aspects of this work, including all used machine learning methods, are described in Section 4. In Section 5, we describe our data processing and both models and computational experiments are introduced. Section 6 contains our results, including a discussion of them, and finally we conclude in Section 7.

2. Background

The purpose of this chapter is to provide the basic understanding needed to read, digest and evaluate the following sections of this thesis. In order to do so, we will first introduce the notion of a severity of illness score and then discuss basic aspects related to machine learning. The section is concluded by combining these two topics with a brief motivating discussion of the use of machine learning for developing better severity of illness scores.

2.1 Traditional acuity scores

Intensive Care Units (ICUs) are departments of hospitals in which severely-ill or -injured patients are provided with life-saving treatment. These patients require continuous monitoring and constant care in order to make sure that deterioration in patient condition is detected early and treated accordingly. As a consequence of the monitoring, ICUs are data rich environments enabling the collection of electronic health records. These records can then be utilized in various different ways.

In ICU settings, an estimate of the *deterioration risk of a patient*, or an *outcome prediction*, is a highly valuable piece of information. In this thesis, the outcome refers to the in-hospital mortality, although in general it could be something more specific, e.g., the dysfunction of a specific organ. Having said that, any such estimates would be useful in triage, resource allocation, in determining appropriate level of care, and in discussions with the patient and the family of the expected outcome of the care [36]. Several traditional acuity, i.e., Severity Of Illness (SOI), scores already exist for these and other purposes in ICUs.

For example, the Acute Physiology and Chronic Health Evaluation (APACHE) scores are general risk-prognostication scores which essentially rely on the assumption that the severity of a disease can be quantified with the abnormality of physiologic variables [57, 60]. APACHE IV published in 2006 [62] uses a dataset of 110,558 patients from 104 intensive care or coronary care units of 45 hospitals. It is based on a *multi-variate logistic regression* model with additional spline terms fitted using a set of *covariate*s, or predictors, to obtain an in-hospital mortality risk. The

covariates consists of the acute physiology score, age and admission circumstances amounting to 142 variables out of which the acute physiology score is based on the most abnormal, or deranged, values during the first 24 hours after ICU admission.

Simplified Acute Physiology Scores (SAPSs) are also examples of acuity scores based on abnormality of physiological variables [57, 60]. SAPS 3 published in 2005 [38, 41] produces a mortality prediction based on three subscores comprising patient characteristics, circumstances of admission and acute physiology. Underlying the model, is a database of 19 577 patients admitted to 307 ICUs around the world. Unlike in APACHE, the subscores are solely based on data available within one hour of ICU admission. SAPS 3 is the arithmetic sum of the subscores, it attains values from 0 to 217, and provides global as well as regional mortality risks through separate logistic regression models. It is noteworthy that one-half of the predictive power of the model is related with the information that is available before ICU admission. Out of the remaining part, 22.5% is related to the circumstances of the ICU admission, and 27.5% to the physiologic data.

Unfortunately, acuity scores such as APACHE are not sufficiently accurate for predicting an individual patient's outcome [34]. This has had the consequence that the focus has shifted into using them, e.g, for comparing ICUs at an organizational level. Moreover, the acuity scoring systems have the tendency to deteriorate as time goes by [40]. The reasons for this are the changes in the lifestyles of the population, changes in the way healthcare is organized and delivered, and in the way diseases are diagnosed, prevented and treated [40]. Even the latest versions of APACHE and SAPS require constant updates and regional customization to remain performant [53].

2.2 Concepts of machine learning

In the introduction, we explained that machine learning allows computers to learn from data. Here, we will briefly introduce some basic machine learning concepts so that the reader can fluently follow the subsequent sections. The content of this section relies on [42, 23].

Usually machine learning is divided into three main categories known as *supervised learning*, *unsupervised learning* and *reinforcement learning*. In this thesis, we will focus on supervised learning, in which a machine learning algorithm uses labeled data to guide the learning process, which results to a trained machine learning model. Two most common categories of supervised learning tasks are classification and *regression* in which the labels are finitely discrete and continuous, respectively. Moreover, since we are interested in the in-hospital mortality predic-

tion task in this work, we have a binary classification task with a patient outcome after an encounter being binary, namely either alive or dead. A machine learning model for classification is called a *classifier* and classifiers can be further divided into decision-rule based and probabilistic classifiers. In this thesis, we are interested in probabilistic classifiers, more of which in Section 4.2, with which we aim to learn a mapping from measurements of patient condition to the probability of dying during some time-span following the measurements. Typically, these measurements are referred to as the predictor variables, or *features*, while the label is called the *target variable*. We will come back to the chosen predictors as well as to the labeling process in more detail in Section 5.1. In addition to the model and algorithm, the training process further requires the notion of a *loss*. In our case, the specific loss is model dependent, but it is an optimization target which takes the true label and the predicted probability as its inputs and produces a number.

In most cases, the learning algorithm tries to minimize the loss on a dataset used for training. However, instead of learning a model that performs well on this training data, the objective of machine learning is to learn a model that performs well on unseen data. Mathematically, this means that we wish to minimize the expectation of the loss with respect to the data generating distribution. In order to achieve this in practice, it is a common approach to perform a three-way split of the available dataset into a *training*, *validation* and *test set*s, cf. Section 5.3. Ultimately, our practical goal is to maximize the performance of the model on the test set, which is the empirical version of minimizing the aforementioned expectation. In order to avoid leaking information about the test set to the model, which could lead the model learning the specifics of the test set, the strategy is to minimize the empirical loss on the training set while monitoring the performance of the model on the validation set. The main uses of the validation set monitoring is to choose the *hyperparameters* of the model, which are parameters of the model not optimized by the learning algorithm, or to stop the learning process before the validation loss starts to degrade. So-called *regularization* parameters are an important class of hyperparameters which can control the amount of *under-* or *overfitting* to the training set. In simple terms, underfitting refers to the model learning a too crude and overfitting a too complex mapping between the predictors and the labels. Both can be diagnosed by comparing the training and validation errors. Typically, underfitting is associated with a high training and validation error while overfitting results in a small training but a high validation error. In both cases, the objective, which is called the *generalization error*, measured empirically by the test set error is high. Regularization aims at balancing between under- and overfitting to produce an optimal model by trading one with another.

Up to this point, we have been discussing the training, validation and test set losses or errors. For one, another way to choose the hyperparameters, or in general to deal with the *model selection problem*, see Section 4.3, is to perform *Cross-Validation* (CV). A common form of cross-validation is known as K-fold CV. In it, instead of a single training and validation set, multiple training and validation sets are formed by partitioning the available dataset, excluding the test set, into K folds and using each fold at a time as a validation set with the remaining folds acting as the training set. Usually, the resulting validation set errors are averaged to obtain a more robust error estimate which can be used to monitor the training similarly as in the case of a single validation set. For another, we have so far talked about the error in its intuitive sense. In this work, we are primarily concerned with the *error metrics* for classification known as the *Area Under the Operating Characteristic Curve* (AUROC) and *Area Under the Precision-Recall Curve* (AUPRC). More about these metrics can be found in Section 4.5. For the purpose of the next chapter, it is sufficient to say here that both metrics are bound by one from above, and the higher the value of the metric, the better the model performs.

2.3 Machine learning and acuity scores

More than a decade has passed since APACHE IV and SAPS 3 were introduced. Like their predecessors, the next generation of ICU severity of illness scores have the opportunity to take advantage of the ever larger and more fine-grained EHRs. The advances in machine learning have, at the same time, made models using these more granular health records possible [28]. In fact, there is evidence that machine learning techniques have performance gains over the more traditional statistical methods for the outcome prediction [30]. In addition to providing more flexible models for describing the non-linear relationship between the predictor variables and acuity, the machine learning models also make real-time predictions possible [27, 28, 31]. Instead of a severity of illness prediction at admission offered by the traditional APACHE and SAPS scores, a real-time prediction is valuable for enabling more frequent individual care decisions [27]. Such models could be used as early warning systems [34] at ICUs [3].

A *telehealth ICU* provides an important example for the need of better and more frequently available severity of illness scores. Such an ICU is a centralized model of care. In it, patients are monitored remotely continuously and, when needed, both consultations and alerts are provided [47]. Via telehealth ICUs, a single physician can cover many care centers, thus increasing efficiency and cutting staffing costs [22]. Although there are conflicting conclusions about the care re-

sults, several studies do show that these programs decrease mortality, shorten ICU and hospital length of stays, and increase utilization of best care practices [22, 32]. Development of machine learning based *Clinical Decision Support Systems* (CDSSs) for Telehealth ICUs is not easy but it is facilitated by the exponentially increased amount of medical data in electronic health records.

Telehealth ICUs have given birth to large deidentifiable data sets such as the eICU Collaborative Research Database (eICU-CRD) [47]. The access to such large, comprehensive, heterogeneous, and granular data sets enables the development of generalizable medical machine learning applications [32]. Specifically, in the ICU setting, generalizability typically refers to the property that the models trained on some hospitals ICU data perform well when deployed at an ICU of an unseen hospital. In particular, this performance should be contrasted against the performance observed for external data of the same hospitals ICU used for training. It is important to study how models generalize since this is essentially how machine learning based CDSSs are deployed to hospitals in practice.

3. Prior work

Theses contain work the foundations of which lie in several prior works. Here, our intention is to highlight the prior work through some examples and by focusing on four categories which we have identified and deemed relevant. Firstly, we focus on the prior work related to mortality prediction machine learning models at ICUs, and secondly, we look into the use of special types of machine learning models, namely deep learning models, in the *pediatric ICU* setting. Going through these examples is relevant for the task of mortality prediction and for constructing and evaluating a deep learning model which are both tasks we cover in this thesis. The aforementioned two categories are then complemented with a survey of prior work related to the generalizability of machine learning models when trained with selected hospitals' data and tested on other hospitals. Finally, we will explore the literature concerning the use of transfer learning methods in the two-center situation. Also, these two latter categories are directly connected to the research questions of this thesis concerning the multi-center generalizability and knowledge transfer.

3.1 Mortality prediction

In this section, we provide a short literature survey of mortality prediction in ICUs. Instead of an extensive survey of the developments in this task, we will focus on practices, datasets and results of a few selected example papers. This will exclude topics such as using textual notes as predictors and development of novel models. What will be included is model calibration, customized models and real-time predictions.

Hug and Szolovits study the feasibility of real-time mortality prediction in ICU environment in [27]. The authors use the Medical Information Mart for Intensive Care (MIMIC) II dataset. This dataset contains medical ICU, critical care unit and surgical ICU data from the Beth Israel Deaconess Medical Center Boston between 2001 and 2007. From MIMIC II, they choose both real-valued (e.g., vital signs) and categorical (e.g., ICU service type) nurse-charted observations as well

as intravenous medications, input/output variables and demographic variables as their predictors. In addition, they use meta and derived variables based on the direct MIMIC II variables. Hug and Szolovits take the mortality in ICU or within 30 days of ICU discharge as their predicted variable. The authors model the predicted variable using logistic regression accompanied with *variable pre-selection* and *backward elimination* [23]. They measure the *discrimination* power of the mode using the Area Under Receiver Operating Curve, see Section 4.5, and the model *calibration* using the *Hosmer-Lemeshow goodness-of-fit*. We note that discrimination refers to the models ability to distinguish between the two output classes of a binary classification task. Furthermore, calibration refers to the ability of the model to produce reliable risk estimates, that is, ones that are consistent with the observed risks.

Hug and Szolovits use so-called pseudo-SAPS II scores for the MIMIC-II patients as their baseline method. Here, the term pseudo comes from neglecting some SAPS II indicators while taking others into account. In addition to their main contribution, a Real-time Acuity Score (RAS), they investigate the Stationary Daily Acuity Score (SDAS), which uses daily summary data, and the Daily Acuity Scores (DASn), which uses daily summary data for individual days n.

In particular, the RAS model has a 3 day AUROC of 0.878. The authors conclude that real-time acuity scores can achieve the discrimination performance of daily acuity scores and superior performance over pseudo-SAPS II. Moreover, automatic variable selection out of hundreds of candidates can give several significant variables for mortality prediction.

In [5], Celi *et al.* aim to show that models customized to specific patient groups outperform gold standard scoring systems. In particular, they focus on three subsets of patients: those diagnosed with acute kidney injury, subarachnoid hemorrhage and elderly patients undergoing cardiac surgery.

We will focus on the case of the acute kidney injury. The authors extract from the MIMIC database the patients with ICD-9 diagnosis of the acute kidney injury. With 1400 patients and a *mortality prevalence* of 30.7%, they aim to predict the in-hospital mortality. They include demographic as well as physiologic variables as their predictors. The physiological predictors are obtained by taking the minimum, maximum, standard deviation and mean value over the first three days in the ICU. Celi *et al.* investigate the performance of three models: logistic regression, *Bayesian network* and *artificial neural network* [42, 23]. They use a correlation-based feature subset algorithm for feature selection. The authors take the AUROC and Hosmer-Lemeshow goodness-of-fit statistic as their evaluation metrics for discrimination and calibration.

The authors report that the artificial neural network performs the best for the

acute kidney injury dataset achieving an AUROC of 0.875. This should be contrasted with the AUROC value of 0.642 of the baseline SAPS score. In the case of subarachnoid hemorrhage and elderly patients in cardiac surgery, the authors report qualitatively similar results with the customized machine learning models outperforming the standard scores in terms of discrimination. In terms of calibration, the models either improved a poorly calibrated gold standard or preserved a well-calibrated gold standard.

In conclusion, Celi *et al.* state that instead of aiming to build models with good performance on a heterogeneous patient population, it is worthwhile to consider customized models trained with specific patient subsets.

Cosgriff *et al.* hypothesize in [9] that sequential modeling can be used to improve the probability calibration of severity of illness scores. In particular, they consider two logistic regression models, the first assigning a risk score, and the second quantifying this risk for the high risk patients.

The authors use the eICU Collaborative Research Database (eICU-CRD). They choose the cohort by requiring the patients to have the APACHE IVa variables and score, age at least 16 years, no readmissions or ICU transfers, minimum/maximum length of stay of 4 hours/365 days, patients with burns or non-renal, non-hepatic transplant are excluded, and vital sings, laboratory results and treatment documentation were required to have at least one recording. These criteria result in the final cohort consisting of 134 946 patient unit stays. Cosgriff *et al.* chose a set of APACHE IVa features, vital signs and laboratory test results are their predictors. For the vital signs, the mean over values during the first 24 hours was recorded, and for the laboratory results, either the minimum or maximum, depending on the test, was taken into account for the same period of time.

Cosgriff *et al.* evaluate a total of five models in their work. The first three are baseline models: the APACHE IVa model, a logistic regression model, and a *Gradient Boosting Machine* (GBM). Their sequential models operate such that the first logistic regression model makes a risk prediction. If the risk is above a threshold then the second logistic regression model, trained on the high-risk patients alone, makes the final mortality prediction. They use *reliability curve*s to evaluate the calibration of the models while the *receiver operating characteristic* and *precision-recall curve*s, as well as the area under these curves, are used to measure the discrimination.

For discrimination, the authors find AUROCs and *Average Precision*s (APs) of 0.864 and 0.460, 0.887 and 0.529, and 0.907 and 0.596 for the APACHE IVa, logistic regression, and gradient boosted trees, respectively. The AUROC and AP for the sequential models with thresholds 0.1 and 0.5 are equivalent to the logistic

regression model. In terms of calibration, the logistic regression outperforms the poorly calibrated APACHE IVa while the sequential models are better calibrated than the baseline logistic regression.

Cosgriff *et al.* conclude by stating that strategies for calibrating models across the risk spectrum are needed, and that the proposed sequential method is such a strategy although provides only a modest improvement.

We have examined three works which take mortality prediction as an example task to address different research questions. This setup does not facilitate a reasonable direct comparison of the research results or conclusions. However, what we have learned from this is the rough content of the two individual most relevant datasets MIMIC and eICU-CRD. Morever, the articles have a lot in common in the way that the data is preprocessed and features are selected, there are similarities with the modeling approaches, and the results are evaluated in the same way using AUROC for the discrimination. These learnings play a role also for the results of this thesis by contributing to the required background for building machine learning models. In our discussion, we will also refer to the quantitative results of Cosgriff *et al.*, in particular, in relation to our own results. Still, it should be said already here that the actual numbers, e.g., AUROCs, are dependent on the dataset setup and labeling. This should always be kept in mind when performing such quantitative comparisons.

3.2 Deep learning for intensive care units

One of the aims of this work is to investigate the applicability of deep learning for in-hospital mortality prediction of ICU patients. For this purpose, in the following, we will examine two example papers concerning applications of deep learning at ICUs. Our focus is on the modeling methodology rather that the specific use cases. This is partly due to scarcity of relevant literature for this rather new application area of deep learning.

Lipton, Kale, Elkan and Wetzel study *diagnosis classification* in a Pediatric ICU (PICU) using irregularly sampled clinical measurements [37]. It is the first study which uses *Long Short-Term Memory* (LSTM) networks [25] for *multi-label classification* of multi-variate PICU time-series.

The used dataset is a collection of clinical time series extracted from the EHR system at Children's Hospital LA consisting of 10 401 PICU visits. There are 13 features, e.g., vitals, which are resampled to an hourly rate and aggregated using the mean within each one hour window. *Forward-* and *backward-fill* followed by *imputation* with expert given values are used to handle *missing values*. Each time-

series is associated with labels from 128 most frequent diagnoses.

Lipton *et al.* experiment with three LSTM variants combining use of *dropout*, *target replication* and auxiliary outputs. In addition, they use a *Multi-Layer Perceptron* (MLP) as their strong baseline method. The baseline is trained using both raw and aggregate hand-engineered features. The authors use the micro- and macro-averaged F1-score and area under receiver operating characteristic curve together with precision at 10 as their evaluation metrics. Their best LSTM network uses two layers of 128 memory cells, dropout of probability 0.5 between layers and target replication. It achieves the metrics 0.8560 (micro-AUROC), 0.8075 (macro-AUROC), 0.2938 (micro-F1), 0.1485 (macro-F1) and 0.1172 (prec. at 10).

The authors conclude that target replication helps to improve performance on all metrics, accelerates learning and reduces overfitting while auxiliary outputs improves performance for most metrics and reduces overfitting. Overall, they argue that LSTMs, especially with target replication, succeed in classifying diagnoses for clinical time series data.

In [1], Aczon *et al.* investigate the applicability of *Recurrent Neural Networks* (RNN) to in-hospital mortality prediction at a PICU. In particular, they focus on the value of dynamic predictions on the given task. The authors use EHRs for a single center, i.e., Children's Hospital Los Angeles, obtained between 2002 and 2016. Their final dataset consists of 12 020 patients with 16 559 encounters. The used predictors include static information from demographics to diagnoses as well as irregularly, sparsely and asynchronously sampled dynamic information from physiologic and laboratory measurements to drugs and interventions. The dynamic data was pre-processed with either zero or fill-forward imputation depending on the type of the feature.

Aczon *et al.* chose to represent their data in terms of input vectors containing five group of measurements from the observations to a scalar time interval representing the forecast time. These input vectors were fed to a LSTM network which is a special kind of a recurrent neural network, see. 4.2.3. The outputs of this network were the dynamic probabilities for an ICU death.

The LSTM results were compared against Pediatric Index of Mortality (PIM) 2, Pediatric RISk of Mortality (PRISM) 3, a Logistic Regression (LR) model and a multi-layer perceptron. The models performance was measured using the receiver operating characteristic curve and AUROC. Focusing on predictions at the 12th hour, it is shown that the LSTM model achieves 0.934 AUROC which should be contrasted with the AUROCs 0.888, 0.861, 0.863 and 0.880 of MLP, LR, PIM 2 and PRISM 3, respectively. Furthermore, LSTM networks predictions are shown to improve as the observation time is increased.

In the conclusions, the authors summarize the performance gains of the deep learning approach and state this is due to access to a more extensive set of features as well as to the dynamic integration incorporated into the data.

These two works both operate in the PICU setting and use datasets with the same origin but they face two different prediction tasks. This makes a direct quantitative comparison of the results meaningless. Instead, what we can learn from the featured works is ways in which recurrent neural networks can operate with the irregularly sampled and heterogeneous data that ICUs produce. Moreover, on a more qualitative level both works highlight the success of LSTM networks in their respective tasks. In fact, this is also our research question in the mortality prediction task but in the generalizability framework. Thus the above reviewed results of the authors of [37, 1] place a strong prior on our expectations of the performance of LSTM networks.

3.3 Generalizability between hospitals

The main objective of this work is to investigate the generalizability of selected machine learning models in the multi-center ICU setting. In what follows, we hence review the relevant literature concerning the generalizability of machine learning models between different hospitals. In this review, we have excluded approaches based on *transfer learning*, more specifically on domain adaptation, which are addressed in the following section.

In [29], Johnson, Pollard and Naumann study the generalizability of an in-hospital mortality prediction model to unseen hospitals. In particular, they make two research hypotheses: (i) a model trained on data from one institution does worse when tested in other institutions and (ii) a so-called locally trained model will outperform a transferred model given sufficient data.

Johnson *et al.* conduct their investigation using the eICU Collaborative Research Database. They model in-hospital mortality with logistic regression using AUROC and the *Standardized Mortality Ratio* (SMR) to measure respectively the discrimination and calibration of the model.

In the first experiment, the authors used two approaches to address hypothesis (i). For one, they used cross-validation on the full training set to evaluate the generalization error, and for another, they evaluated the generalization error using held-out individual hospitals. They found that the cross-validation error overestimates the evaluation metrics but not by an impractical amount. In the second experiment, Johnson *et al.* addressed hypothesis (ii). They chose a hospital setting aside 500 of its patients as a test set. The remaining hospitals were left as the

training set. They proceeded iteratively with the remaining patients of the chosen hospital and sampled 200 additional patients at each iteration into the initially empty recalibration set. The authors examined two models: a transferred model trained with the full training set and recalibrated at each iteration with the recalibration set and a local model trained at each iteration with the recalibration set. Johnson *et al.* found that although the local model was well calibrated it could not reach the discriminative ability of the transferred model even with large sample sizes.

Rasmy *et al.* investigate the validity of a machine learning model in a large heterogeneous EHR dataset in [51]. In addition, they study the generalizability of a model trained in one hospital to other hospitals.

They use the Cerner Health Facts EMR data which has nearly 50 million patients across over 600 hospitals. The authors aim to predict the onset of heart failure of which there are more than 150 000 cases in the dataset. After a case-control matching, they obtain 152 790 cases and 1 152 517 controls. Rasmy *et al.* use the REverse Time AttentIoN (RETAIN) model which is a highly accurate, interpretable RNN model with *attention* [8]. This model takes a sequence of visits of a patient containing medical codes as its input and outputs the risk score of heart failure onset. The authors took diagnoses, medication and surgical procedures for the medical codes.

In [51], the authors perform four computational experiments and the generalizability experiments. In particular, in their fourth experiment, they run RETAIN on the full dataset which results into AUROC of 0.822. This is contrasted against 0.766 achieved by the baseline logistic regression model. In the generalizability tests, they consider the ten largest hospitals. RETAIN is trained with each hospitals training set and test with all test sets of the ten hospitals. They also consider the model trained with the training sets of all hospitals. Rasmy *et al.* report that the prediction accuracy varies a lot from hospital to hospital. They find that the AUROC decreases on average by 3.6% when tested on a different hospital. Finally, the authors state that the model trained on all hospitals' data always outperforms the models trained with individual hospitals' data.

In summary, the main contribution of [51] is the validation of the generalizability of a RNN-based model to a large heterogeneous EHR datasets. Generalizability across hospitals and clinics is shown but it is stated that practical applications require additional testing due to the observed great variability between patient groups.

In these two featured works, the authors investigate the generalizability of a machine learning model trained using the data of selected hospitals but eval-

uated on unseen hospitals' data. Clearly the used datasets and prediction tasks are different but nevertheless the qualitative results can be compared. Firstly, both works find that the model's performance degrades when transferred to an unseen hospital. Secondly, both works agree that a transferred model trained on multiple centers' data outperforms a local model trained on data of the hospital under the evaluation. It therefore seems that these observations are both robust but also that they depend on the amounts of data which are available for the transferred and local models. In this thesis, we will revisit both of these questions in our computational experiments. Furthermore, we will also refer to the quantitative results of Johnson *et al.* in the context of discussion of our results.

3.4 Domain adaptation for intensive care units

In this section, we provide a short survey of the domain adaptation literature associated with the two-center ICU setting. We limit ourselves to two centers as we are not aware of works focusing on multiple centers. In addition, this survey excludes approaches based on deep learning [48, 2, 21]. With this restriction in mind, the provided list aims to be more comprehensive than for the other parts of this literature survey section.

Desautels *et al.* study the applicability of transfer learning to a two-center dataset in [13]. That is, they investigate whether using source data from one center can be used to improve the performance of a machine learning model on another center in the task of mortality prediction.

The used source data was drawn from the MIMIC-III database while the target dataset was selected among the 109 521 adult inpatient encounters of the University of California San Francisco hospital system. After applying feature and prediction time based filters, the final source dataset included 39 071 encounters while the target set contained 48 249 patient encounters. The authors chose some vitals, the *Glasgow Coma Scale* (GCS) and some laboratory measurements as their predictors. They applied physiological threshold based *outlier removal* to these features. The data was binned to hourly windows for which each feature was averaged and carry-forward imputation was conducted to impute the missing values. Each input to the predictive model consisted of the age of the patient and a concatenated feature vector of measurements of five consecutive hours before the prediction time. Encounters were binarized according to the discharge status to provide the class labels.

Desautels *et al.* chose a boosted ensemble of *decision tree*s as their machine learning model. Their transfer learning passed both the *source* and *target data* to the

model at training time. The target and source data were provided with different *instance weights* w and $1 - w$ (see Section 4.4.1 for further information). The weight w was used as an experimental parameter. The authors simulated a deployment to a new center by sweeping over the size of the target data used for training while using a 10-fold cross-validation over the target data to provide a testing environment. The results of this setup showed that a transfer learning method with a weight w chosen by a *nested cross-validation* yielded the best performance. The delivered AUROC was 0.7852 and 0.8043 with 0.25% and 0.5% of the target data used for training. This should be compared with AUROCs of 0.4961 and 0.5510 obtained with no source data in the training set. Using only source data yielded an AUROC of 0.7831.

In the discussion part of [13], the authors conclude that their implementation of transfer learning improves mortality prediction over the target-only and the source-only training. They furthermore state that although the source domain classifier may be considered a reasonable choice, transfer learning can provide performance improvements since there are demographic and care practice differences between centers.

In [10], Curth *et al.* analyze the problem of transferring EHR-based clinical prediction models across hospitals and EHR systems. In particular, they discuss the formalization of domain adaptation framework, and provide empirical evaluation of several domain adaptation methods for the task of predicting readmission and mortality after discharge from an ICU.

In regard to the formalization, we only state that Curth *et al.* identify two domain adaptation regimes, i.e., covariate and *concept shift*, and associate a number of methods into these categories. Firstly, they describe a method which assigns instance weights to the source and target domains called a pooled method, cf., Section 4.4.1. Secondly, a sequential modeling approach using a model trained on source data as a prior to regularize the the target model is introduced. Thirdly, the authors describe a hierarchical model which relies on a hierarchy of models for the underlying task, cf., Section 4.4.2.

The authors use a source dataset consisting of VU University medical center (VUmc) admissions gathered between 2004 and 2016. Furthermore, they have two target datasets: the first consisting of new VUmc admissions between 2016 and 2018 (VUmc Epic), and the second comprising data from the Elisabeth-TweeSteden (ETZ) hospital. *Cohort* selection criteria left them with 14 105, 2847 and 13 300 admissions for the source and two target (VUmc Epic, ETZ) datasets. The authors chose patient demographics, admission characteristics, clinical observations, physiological measurements, laboratory studies and medications as their predictors. The

preprocessing of the features included windowing and subsequent aggregations, e.g., average, standard deviation, the first and last values, the minimum and maximum. In order to label their data, the authors identified subsequent admissions or deaths between 12 hours and 7 days after the ICU discharge.

Curth *et al.* chose two base models underlying the domain adaptation methods, namely a logistic regression and gradient boosted tree model. They use ten-fold cross-validation with 10 repeats to obtain AUROCs whose mean is used to describe the performance of the methods. The authors also use a feature selection algorithm.

Two experiments are introduced in the paper to address the posed research questions. First applies the baseline, i.e., source-only and target-only, and domain adaptation methods to the full target datasets while the second measures the performance of the methods when the proportion of target training data is increased gradually. Note that source- and target-only methods use only source and target data in training, respectively. For the second experiment, the authors find that access to source data in addition to target data reduces the amount of target data needed by a well performing model in the target domain. In particular, they observe that the hierarchical model outperforms the pooled model while both are better than the baselines. Furthermore, Curth *et al.* note that the prior method succeeds in improving the target performance without using the source data. Comparing the base models leads to the conclusion that the more advanced machine learning models will benefit more from the domain adaptation methods.

In conclusion, the authors state that when machine learning models are deployed to new hospitals, a large dataset from another hospital can considerably reduce the amount of required target data to reach a good model.

Both of these two-center domain adaptation studies, although they did not use completely identical methods, concluded that domain adaptation improves the performance of a model over the source- and target-only baselines. In particular, this observation holds for the weighted and feature augmentation methods outlined in Sections 4.4.1 and 4.4.2 and used in this thesis. This plausible qualitative result seems to be a robust observation and, in this thesis, we will examine if it holds in the multi-center case. In relation to the prior work on the generalizability, it is further interesting to note that also Desautels *et al.* find that the source-only method performs better than the target-only method. Curth *et al.* instead show that this is target training set size dependent. As noted previously, this is also something that we will investigate in this thesis. Finally, the observation of Curth *et al.* that more advanced methods benefit more from the domain adaptation techniques is perhaps not surprising given that they have more capacity to find useful relations

in the larger amount of data of the pooled methods, for example. Our work relies on this observation as we focus on flexible models from the beginning. However, we do not examine this effect itself any further.

4. Methods

In this chapter, we first introduce the multi-center dataset used to address the research questions of this work. Furthermore, we will discuss the methods used to analyze this dataset. This includes introducing the machine learning models, model selection tools and domain adaptation methods. Finally, we will discuss the metrics used to evaluate the used models.

4.1 Dataset

The eICU Collaborative Research Database (eICU-CRD) is a multi-center intensive care unit database [47]. It consists of electronic health records of over 200 000 severely ill patients that received critical invasive treatment, collectively in over 200 hospitals in the US. The database has been collected from diverse information sources at ICUs in the course of a working telehealth ICU. The eICU-CRD has been archived by Philips Healthcare in the course of the eICU telehealth program and transformed into research use by the Laboratory for Computational Physiology (LCP) at MIT and the eICU Research Institute (eRI). Having said that, this dataset is ideal for our purpose of studying the generalizability of machine learning methods in the multi-center situation. How the dataset is used in this work is explained to the detail in Section 5.1.

4.2 Models

In this section, we introduce three machine learning models which are used in this thesis to predict in-hospital mortality. The first two, that is *random forest* and gradient boosted trees, are tree-based methods relying on *ensemble*s of decision tree models. These are more traditional methods which do not as such take into account the time-series nature of EHR data. In order to consider the time-series nature explicitly, we will further introduce recurrent neural networks and a special version of them known as the long short-term memory network.

4.2.1 Random forest

Decision trees are intuitive and powerful *discriminative*, *non-parametric* classifiers [23, 42]. They produce a recursive binary partitioning of the feature space when trained on a training set. When classifying new instances, they locate the region in which the instance lies and, based on the training set, take a majority vote inside the region to determine the label of the instance. Such decision trees are low *bias*, high *variance* classifiers. This is what makes them good candidates for ensembling. A random forest is an ensemble of decision trees grown using *bootstrap aggregating*, or *bagging*, and random feature selection per split in the tree [23, 42]. In the following, we will examine to some level of detail how random forests actually work.

Let us start with decision trees. In order to do this, we follow mainly [23] but note that other sources and methods are also available. Having said that, the recursive splits which define a decision tree are determined by minimizing a selected loss function. However, minimizing this loss exactly with respect to the split variables and thresholds is an intractable *combinatorial optimization problem*. The solution is to grow the decision tree incrementally using a *surrogate loss function*. In order to define this loss, we first need some preliminary notions. Consider a training set $\{(x_i, y_i)\}_{i=1}^N$ with $x_i \in \mathbb{R}^D$ and $y_i \in [1, K] \cap \mathbb{Z}$. Given a set $R \subset \mathbb{R}^D$, we define the proportion of class k in region R as

$$\hat{p}_k(R) \equiv \frac{\sum_{i=1}^N I[x_i \in R] I[y_i = k]}{\sum_{i=1}^N I[x_i \in R]},$$

where I is the indicator function. In our case, we define a measure of node impurity

$$Q(R) \equiv \sum_{k=1}^K \hat{p}_k(R) \Big(1 - \hat{p}_k(R)\Big),$$

which is just the *Gini-index*. Other choices, like the *miss-classification error* or *cross-entroopy*, are equally-well possible. However, the advantage of the Gini-index over the miss- classification error is that it is differentiable and more sensitive to changes in the node probabilities. The loss function at a split over the region R can be then defined as

$$\begin{aligned}\mathcal{L}_j(s; R) = &\, Q(\{x_i \in R | x_{ij} \leq s, i \in [1, N] \cap \mathbb{Z}\}) \\ &+ Q(\{x_i \in R | x_{ij} > s, i \in [1, N] \cap \mathbb{Z}\}),\end{aligned}$$

where s is a threshold and j is the feature with respect to which the split is made. Note that this loss is to be minimized with respect to j and s. The process of fitting

a decision tree starts by minimization over the entire feature space and repeats recursively by looking at minimization over each of the resulting regions. This process continues until a stopping criterion is met. Usual stopping criterion may be that the node, or region, contains no more than a specified number of training instances. Once a tree has been fitted, a partition $\{R_m | m \in [1, M] \cap \mathbb{Z}\}$ has been obtained. Then, given a new instance x, its class label is given by

$$f(x) = \sum_{m=1}^{M} \arg\max_k \hat{p}_k(R_m) I[x \in R_m] \,,$$

which summarizes how a tree predicts unseen examples. Decision trees have several attractive features. They are interpretable, can handle mixed discrete and continuous features, they are invariant under monotone transformations of the features, they do automatic feature selection, they are relatively robust against outliers, they scale well to large datasets and they can be modified to handle missing inputs [42]. The downsides are that decision trees do not predict very accurately and that they are unstable against small changes in the training data.

Decision trees, when grown sufficiently deep, have a low bias but they are noisy with respect to variations in the training data. Since bootstrapping is meant to average noisy but unbiased models, it is an ideal choice for decision trees. Bootstrapping effectively lowers the variance of the estimator making the ensemble a more robust model. In order to achieve a lower variance, it is also essential that the members of the ensemble are sufficiently decorrelated. In a random forest, this is accomplished by randomly choosing a subset features at each split for the split candidates. Combining these two stages gives rise to the following fitting algorithm [23].

1. For $b \in [1, B] \cap \mathbb{Z}$:

   (a) Draw a bootstrap sample Z of size N from the training data.

   (b) Grow a random-forest tree T_b to bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size $n_{\min}$ is reached.

      i Select p variables at random from the D variables.

      ii Pick the best variable/split-point (j, s) among the p.

      iii Split the node into two daughter nodes.

2. Output the ensemble of trees $\{T_b\}_{b=1}^{B}$

Once a model has been fitted it can be used for classification. If x denotes a new instance then the prediction of the random forest is given by

$$f(x) = \text{mode}\{\hat{c}_b(x)\}_{b=1}^{B},$$

where $\hat{c}_b(x)$ is the prediction of the bth tree. Through bagging random forests overcome the accuracy issue of individual decision trees. However, this comes at the expense of losing some of the interpretability of a decision tree [42].

4.2.2 Gradient boosted trees

Boosting is a powerful ensemble method in which a collection of weak *base learner*s is trained sequentially on modified versions of the data [23, 42]. Gradient boosting is a general form of boosting which enables learning based on generic differentiable loss functions [16]. In the following, we will describe gradient based boosting for trees and restrict our attention to the classification problem.

In boosting, M weak learners $h(x, a_m)$ parametrized by a_m are combined additively with weights β_m to form a *committee* for $m \in [1, M] \cap \mathbb{Z}$. For binary classification, we can write

$$f_M(x) = \sum_{m=1}^{M} \beta_m h(x, a_m),$$

where the output $f_M(x)$ gives the class labels $\{\pm 1\}$ via the sign function $\text{sgn} f_M(x)$. The task is then to optimize the parameters β_m and a_m for all $m \in [1, M] \cap \mathbb{Z}$. The full optimization problem

$$\min_{\{\beta_m, a_m\}_{m=1}^{M}} \sum_{i=1}^{N} \mathcal{L}\Big(y_i, \sum_{m=1}^{M} \beta_m h(x_i, a_m)\Big)$$

is however usually intractable. Forward stage-wise modeling is its *greedy* substitute consisting of an initialization $f_0 = 0$ and the iterative steps

$$\begin{aligned}(\beta_m, a_m) &= \arg\min_{\beta, a} \sum_{i=1}^{N} \mathcal{L}\Big(y_i, f_{m-1}(x_i) + \beta h(x_i, a)\Big), \\ f_m(x) &= f_{m-1}(x) + \beta_m h(x, a_m),\end{aligned}$$

for $m \in [1, M] \cap \mathbb{Z}$. For the exponential loss this leads to the well-known AdaBoost algorithm while the squared loss gives rise to the least squared boosting [23, 42]. Other, more robust loss functions are more difficult to optimize. It is possible to overcome these difficulties by contrasting the forward stage-wise modeling with

the method of steepest descents. For finite training sets, the steepest descents minimizing a loss is defined with initial guesses f_{i0} by

$$\begin{aligned} f_{im} &= f_{i\,m-1} - \rho_m g_{im}\,, \\ g_{im} &= \left.\frac{\delta \mathcal{L}(y_i, f(x_i))}{\delta f(x_i)}\right|_{f(x_i)=f_{i\,m-1}}\,, \\ \rho_m &= \arg\min_{\rho} \sum_{i=1}^{N} \mathcal{L}(y_i, f_{i\,m-1} - \rho g_{im})\,, \end{aligned}$$

for $i \in [1, N] \cap \mathbb{Z}$ and $m \in [1, M] \cap \mathbb{Z}$. While, on its own, the steepest descents does not constitute to a useful approach, it can be combined with the forward stage-wise modeling by noticing that $h(x_i, a_m)$ plays a role similar to the gradient $-g_{im}$. The resulting gradient boosting algorithm

$$\begin{aligned} a_m &= \arg\min_{a} \sum_{i=1}^{N} \Big(- g_{im} - h(x_i, a) \Big)^2\,, \\ \rho_m &= \arg\min_{\rho} \sum_{i=1}^{N} \mathcal{L}(y_i, f_{i\,m-1} - \rho_m h(x_i, a_m))\,, \\ f_m(x) &= f_{m-1}(x) + \rho_m h(x, a_m)\,, \end{aligned}$$

is based on fitting a base learner $h(x, a)$ to the negative gradients. The main difference to the method of steepest descents is that the components $(h(x_1, a_m), \ldots, h(x_N, a_m)$ are not independent but set to be the predictions of a base learner.

If the base learners are chosen to be J_m-terminal node decision trees then $a_m \equiv \{\gamma_{jm}, R_{jm}\}_{j=1}^{J_m}$ and the generic gradient boosting algorithm attains the following form [16, 23].

1. Initialize $f_0(x) = \arg\min_{\gamma} \sum_{i=1}^{N} \mathcal{L}(y_i, \gamma)$
2. For $m = 1$ to M:
   (a) Compute
   $$r_{im} = -\left.\frac{\partial \mathcal{L}(y_i, f(x_i))}{\partial f(x_i)}\right|_{f=f_{m-1}}, \forall i \in [1, N] \cap \mathbb{Z}$$
   (b) Fit a regression tree to the targets r_{im} giving terminal regions R_{jm} for $j \in [1, J_m] \cap \mathbb{Z}$
   (c) For $j = 1$ to J_m compute
   $$\gamma_{jm} = \arg\min_{\gamma} \sum_{i=1}^{N} I[x_i \in R_{jm}] \mathcal{L}(y_i, f_{m-1}(x_i) + \gamma)\,,$$

(d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I[x \in R_{jm}]$

3. Output $f(x) = f_M(x)$

Furthermore, if we relax the assumption of two classes, then the gradient boosting algorithm for K classes is given as follows [16, 23].

1. Initialize $f_{k0}(x) = 0$ for all $k \in [1, K] \cap \mathbb{Z}$

2. For $m = 1$ to M:

   (a) Set

   $$p_k(x) = \frac{e^{f_k(x)}}{\sum_{k'=1}^{K} e^{f_{k'}(x)}}, \forall k \in [1, K] \cap \mathbb{Z}$$

   (b) For $k = 1$ to K:

   i Compute $r_{ikm} = y_{ik} - p_k(x_i)$ for all $i \in [1, N] \cap \mathbb{Z}$

   ii Fit a regression tree to the targets r_{ikm} giving terminal regions R_{jkm} for $j \in [1, J_m] \cap \mathbb{Z}$.

   iii Compute

   $$\gamma_{jkm} = \frac{K-1}{K} \frac{\sum_{i=1}^{N} I[x_i \in R_{jkm}] r_{ikm}}{\sum_{i=1}^{N} I[x_i \in R_{jkm}] |r_{ikm}| \left(1 - |r_{ikm}|\right)}, \forall j \in [1, J_m] \cap \mathbb{Z}$$

   iv Update $f_{km}(x) = f_{k,m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jkm} I[x \in R_{jkm}]$

3. Output $f_k(x) = f_{kM}(x)$ for all $k \in [1, K] \cap \mathbb{Z}$

4.2.3 Recurrent neural networks

Neural networks are *parametric* models which have achieved a great success especially during the last ten years. This success has been due to major advances in, e.g., optimization techniques, due to development of computational resources and also due to increase in the amount and availability of data. Recurrent neural networks are a special kind of networks designed to work with sequence data [19, 18]. They can scale to long sequences and handle sequences of varying lengths due to parameter sharing. Recurrent neural network can also handle varying types of inputs and outputs using one-to-many, many-to-one or many-to-many architectures. In the following, we will focus on the many-to-many, i.e., sequence-to-sequence, architecture and to a specific type of RNN known as the long short-term memory network [25, 19, 18].

Let us consider the task of sequence-to-sequence classification. That is both inputs $(x^{(t)})_{t=1}^{T}$ and categorical outputs $(y^{(t)})_{t=1}^{T}$ are given for the full T-term sequence. Then, for a basic RNN, the forward equations read

$$\begin{aligned} a^{(t)} &= b + Wh^{(t-1)} + Ux^{(t)}\,, \\ h^{(t)} &= \tanh(a^{(t)})\,, \\ o^{(t)} &= c + Vh^{(t)}\,, \\ \hat{y}^{(t)} &= \mathrm{softmax}(o^{(t)})\,, \end{aligned}$$

where $a^{(t)}$ denotes the activation, $h^{(t)}$ the hidden state, $o^{(t)}$ the output and $\hat{y}^{(t)}$ the prediction at time t. Moreover, b and c denote bias vectors while W, U and V are weight matrices of the cell. This recurrent structure is illustrated in Figure 4.1.

Unfortunately, the basic RNN suffers from the problem of vanishing or exploding gradients [18]. For simplicity, if we consider a simple linear recurrent connection without biases and inputs then

$$h^{(t)} = Wh^{(t-1)} = W^t h^{(0)}$$

such that we have the decomposition

$$h^{(t)} = Q\Lambda^t Q^T h^{(0)}$$

when the eigendecomposition $W = Q\Lambda Q^T$ exists. When t increases, only the component of $h^{(0)}$ in the direction of the eigenvector with the largest eigenvalue survives. If the corresponding eigenvalue is below zero, the hidden state, and with it its gradient, tends to vanish. In turn, for eigenvalues above zero, the opposite happens and the result increases without bound.

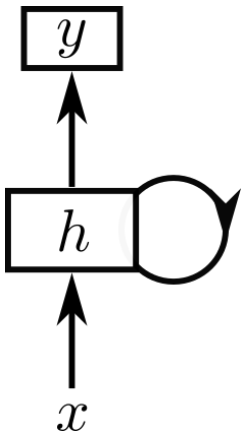

Figure 4.1: Graphical illustration of a simple sequence-to-sequence RNN model. Here, x, y and h refer to the input, output and hidden unit, respectively. The looping arrow denotes the recurrent connection.

So-called gated RNNs are based on the idea of creating paths through time that have gradients that do not vanish or explode. A long short-term memory network is an example of a gated RNN. For a LSTM cell, the forward equations are given by

$$
\begin{aligned}
f^{(t)} &= \sigma(b^f + U^f x^{(t)} + W^f h^{(t-1)}\,,\\
s^{(t)} &= f^{(t)} s^{(t-1)} + g^{(t)} \sigma(b + U x^{(t)} + W h^{(t-1)})\,,\\
g^{(t)} &= \sigma(b^g + U^g x^{(t)} + W^g h^{(t-1)})\,,\\
q^{(t)} &= \sigma(b^q + U^q x^{(t)} + W^q x^{(t-1)})\,,\\
h^{(t)} &= \tanh(s^{(t)}) q^{(t)}\,,
\end{aligned}
$$

where $f^{(t)}$ is the forget gate, $s^{(t)}$ is the cell state, $g^{(t)}$ is the external input gate and $q^{(t)}$ is the output gate. In addition, b^f, b^g and b^q are the corresponding bias vectors with U^f, W^f, U^g, W^g, U^q and W^q being similarly the weight matrices. These cell equations are shown graphically in Figure 4.2. It is the gates which enable LSTM states to store and access information even over long periods. This is mitigating the vanishing and exploding gradients problem [19].

Recurrent neural networks are usually trained by gradient-based optimization algorithms. In the simplest case, Stochastic Gradient Descent (SGD) is used. SGD algorithm is based on the iterative weight update rule

$$\theta_{n+1} = \theta_n - \alpha \nabla_{\theta_n} J(\theta_n)\,,$$

where θ_i denotes all network weights at iteration i and α is an empirically chosen learning rate. At each iteration, the expectation in the loss function

$$J(\theta) = \mathrm{E}_{(x,y)}[\mathcal{L}(y, f(x;\theta))]\,,$$

where $f(x;\theta)$ denotes the output of the network, is approximated by sampling a single element of the training set [18]. The gradient

$$\nabla_\theta J(\theta) = \mathrm{E}_{(x,y)}[\nabla_\theta \mathcal{L}(y, f(x;\theta))]\,,$$

is typically computed by an algorithm known as *Back-Propagation Through Time* (BPTT) which is just automatic differentiation in the reverse mode [18]. Several deep learning frameworks including PyTorch, which is used in this work, realize BPTT, see Section 5.2.3 for more details.

4.3 Model selection

Model selection refers to the task of determining an optimal model which is used for the prediction task. Typically, and this holds for the models of Section 4.2,

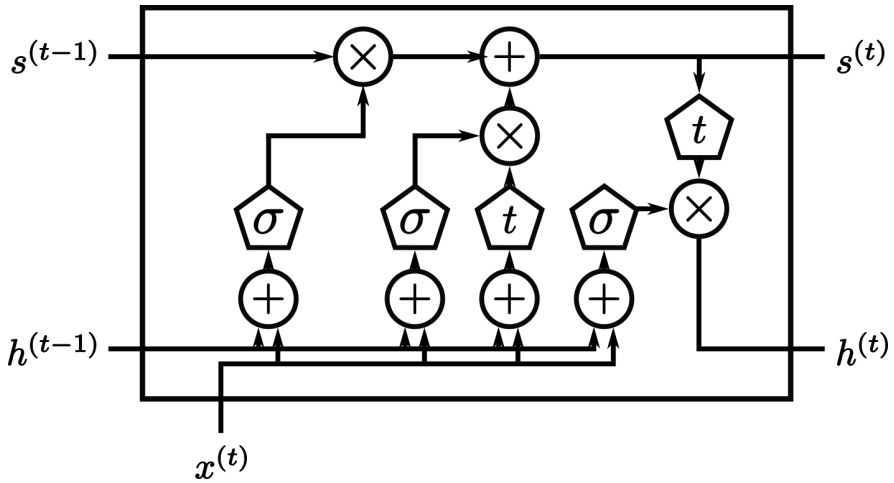

Figure 4.2: Graphical illustration of a LSTM cell. Circles with plus/multiplication signs denote elementwise addition/multiplication. Pentagons denote activation functions with σ and t referring to sigmoid and hyperbolic tangent, respectively.

a model family comes with a set of hyperparameters which need to be chosen prior to the model fitting process. Different hyperparameters give rise to different models within the model family and we need methods for choosing between these different models. In this work, this choice is made based on so-called *Bayesian optimization* technique which is the topic of the following section.

4.3.1 Bayesian optimization

Bayesian optimization is an optimization technique useful specifically for optimizing expensive *black-box functions* globally [4, 55]. That is, it is useful for solving

$$\max_{x \in A} f(x)\,,$$

where it is usual to assume that $f : \mathbb{R}^d \to \mathbb{R}$ is continuous and the feasible set $A \subset R^d$ is compact. Moreover, by an expensive black-box function, it is usually meant that the objective is expensive to evaluate, it is non-convex and derivative information is not available. In the context of machine learning, such settings usually apply to the model selection problem. Optimizing the hyperparameters of a machine learning model is typically expensive since each evaluation of the scoring function requires one to fit and evaluate the model. If the evaluation is based on cross-validation then these tasks have to be done multiple times. It is also clear that this optimization problem can have multiple local solutions and that gradients are not generally available. Standard Bayesian optimization works sequentially in two steps: firstly, it builds a surrogate function to f, and secondly, uses an *acquisition function* to explore the domain, see Figure 4.3. In what follows, we will briefly

introduce how these steps are realized in practice.

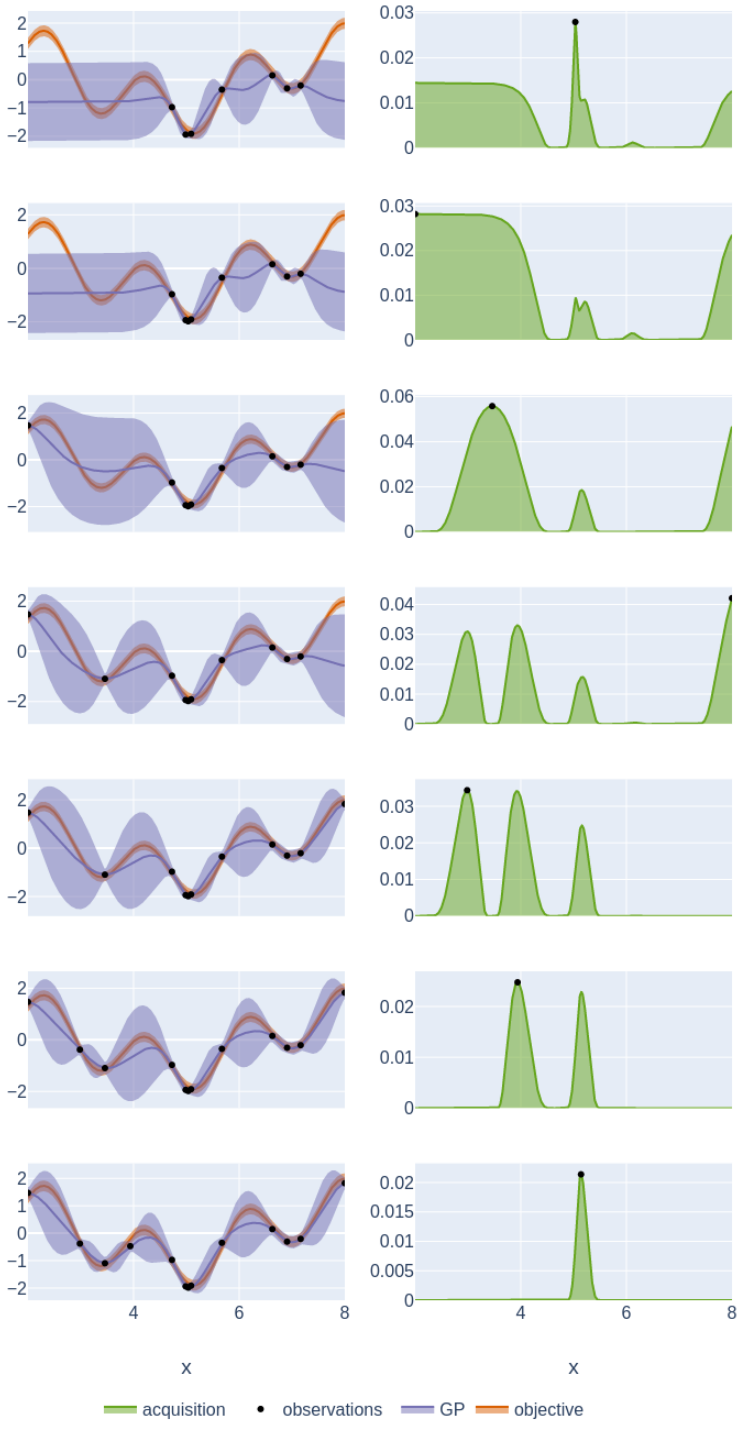

Figure 4.3: Illustration of Bayesian optimization in action. Optimization iterations proceed from top to bottom. On the left, we show a noisy objective function (red), Gaussian process surrogate function (blue) and observed points (black). On the right, we show the acquisition function (green) together with its maximum corresponding to the next observation (black).

Typically, the surrogate function comes from the family of functions representable by a *Gaussian process*. A Gaussian process is a stochastic process with the set of reals as its index set such that every proper subset follows a multivariate normal distribution [50]. Such process can be uniquely defined by specifying a mean function $\mu_0 : \mathcal{X} \to \mathbb{R}$ and a covariance function $k_0 : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. In the presence of n noisy observations $y = (y_1, \dots y_n)^T$ of the objective function f at $x = (x_1, \dots, x_n)^T$ for $x_i \in \mathcal{X} \subset R^d$, a Gaussian process model can be written as

$$\begin{aligned} f|x &\sim \mathrm{MVN}(m, K)\,, \\ y|f &\sim \mathrm{MVN}(f, \sigma^2 \mathbb{1})\,, \end{aligned}$$

where $f = (f_1, \dots, f_n)^T$, MVN denotes a multivariate normal distribution,

$$\begin{aligned} m &= (\mu_0(x_1), \dots, \mu_0(x_n))^T\,, \\ K &= \begin{pmatrix} k_0(x_1, x_1) & \dots & k_0(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k_0(x_n, x_1) & \dots & k_0(x_n, x_n) \end{pmatrix}\,, \end{aligned}$$

σ^2 denotes the variance of the observations and $\mathbb{1}$ is the identity matrix. Usually one considers stationary Gaussian processes for which m_0 is a constant function, typically zero, and the covariance kernel specifies the properties of the surrogate function. A popular choice for the covariance function is the squared exponential function

$$k_{\mathrm{SE}}(x, x') = \theta_0 \prod_{i=1}^{d} \exp\left(-\frac{(x_i - x_i')^2}{2\theta_i} \right)\,,$$

which is parametrized by θ_0 and the length scales $(\theta_1, \dots, \theta_d)^T$. In Bayesian optimization, it is common to determine the hyperparameters θ of the Gaussian process by maximizing the likelihood of the data given by

$$\begin{aligned} \log p(y|x, \theta) = &-\frac{1}{2}(y - m)^T \left(K + \sigma^2 \mathbb{1} \right)^{-1} (y - m) \\ &- \frac{1}{2} \log \left| K + \sigma^2 \mathbb{1} \right| - \frac{n}{2} \log(2\pi)\,. \end{aligned}$$

Alternative to this would be to define hyperpriors for a fully Bayesian treatment [56] but this is not covered here. Owing to the properties of Gaussians, also the posterior distribution is a Gaussian given by

$$f_{n+1}|x_{n+1}, \mathcal{D}_n \sim \mathcal{N}(\mu_{n+1}, \sigma_{n+1}^2)\,,$$

where

$$\mu_{n+1} = \mu_0(x_{n+1}) + k_{n+1}^T \Big(K + \sigma^2 1\Big)^{-1} (y - m)\,,$$
$$\sigma_{n+1}^2 = k(x_{n+1}, x_{n+1}) - k_{n+1}^T \Big(K + \sigma^2 1\Big)^{-1} k_{n+1}\,,$$

while $k_{n+1} = (k(x_{n+1}, x_1), \ldots, k(x_{n+1}, x_n))^T$ and $\mathcal{D}_n = \{(x_i, y_i) : i \in [1, n] \cap \mathbb{Z}\}$. The posterior distribution is the ingredient needed in Bayesian optimization as it provides information about the next potential point to look at when aiming for the optimal solution.

In addition to the surrogate function, the acquisition function needs to be chosen to carry out Bayesian optimization. Its function is to guide the search for the optimum [4]. Here, we will only look at an acquisition function known as the *expected improvement*. The expected improvement acquisition function for the next point x_{n+1} is defined as

$$\alpha_{\mathrm{EI}}(x_{n+1}; \mathcal{D}_n) = E_{f_{n+1}|x_{n+1},\mathcal{D}_n}[(f_{n+1} - \tau) I(f_{n+1} > \tau)]\,,$$

where I is the indicator function and τ is an incumbent target. In addition to the probability, the expected improvement also takes into account the magnitude of improvement unlike, e.g., the probability of improvement [4, 55]. Hence, it provides naturally a compromise between *exploration* and *exploitation*. Since the posterior is Gaussian, the expected improvement can be reduced to

$$\alpha_{\mathrm{EI}}(x_{n+1}; \mathcal{D}_n) = \Big(\mu_{n+1} - \tau\Big)\Phi\left(\frac{\mu_{n+1} - \tau}{\sigma_{n+1}}\right) + \sigma_{n+1}\phi\left(\frac{\mu_{n+1} - \tau}{\sigma_{n+1}}\right),$$

where ϕ and Φ are the normal probability density and cumulative density functions, respectively. In practice, the target τ is unknown but it is set to

$$\tau = \max_{i \in [1,n] \cap \mathbb{Z}} y_i\,,$$

to emulate the best observed value [55]. Finally, the best informed guess of the optimal point is given by

$$x_{n+1} = \arg\max_x \alpha_{\mathrm{EI}}(x; \mathcal{D}_n)\,,$$

that is, by optimizing the acquisition function.

4.4 Domain adaptation

In several machine learning applications, the target and source data distributions or tasks differ in some way. In such cases, usually, there is plenty of available labeled source data $\{(x_i^{\mathrm{src}}, y_i^{\mathrm{src}}) : i \in [1, N_{\mathrm{src}}] \cap \mathbb{Z}\}$ but only scarcely labeled target

data $\{(x_i^{\text{trg}}, y_i^{\text{trg}}) : i \in [1, N_{\text{trg}}] \cap \mathbb{Z}\}$. It can also very well be that only unlabeled target data $\{x_i^{\text{trg}} : i \in [1, N_{\text{trg}}] \cap \mathbb{Z}\}$ is available. In either situation, a question arises how to utilize the available data for training a machine learning model which generalizes on the target distribution and task. Such a question can be addressed with transfer learning [61]. If the target and source tasks are the same and only the *feature space*s and *marginal distribution*s, i.e., domains, differ then one typically refers to domain adaptation. Furthermore, in this work, we have domains which differ only by their marginal distributions which is called homogeneous domain adaptation. In fact, in our work, there are multiple domains $\{(x_i^s, y_i^s : i \in [1, N_s] \cap \mathbb{Z}\}$ for $s = 1, \ldots, S$, each corresponding to a single hospital, which calls for multi-source domain adaptation methods [58]. In what follows, we survey two domain adaptation methods which have multi-source extensions and which are mostly model agnostic.

4.4.1 Weighted method

In the case of a single source domain, it is possible to use instance weights in a simple way to cope with the domain adaptation situation [11]. In this approach, for example, the source instances are uniformly re-weighted to either decrease or increase their importance relative to the target instances. The idea behind this is that typically there are much fewer target instances than source instances and without re-weighing the guiding effect of target instances is washed out by the source instances. This is a simple *supervised domain adaptation* method. Note that several models can take advantage of weighted instances during the training process and that this method can be also applied in the multi-source situation by pooling the source domains together.

4.4.2 Feature augmentation method

In the *feature augmentation method* (FAM) [11], as its name implies, one augments the design matrix with additional features. This method is an example of a supervised domain adaptation method in which limited labeled target data is available. In the

simplest case of a single source domain, the augmented feature vectors are

$$x \xrightarrow{x \in X_{\text{src}}} \begin{pmatrix} x \\ x \\ 0 \end{pmatrix},$$
$$x \xrightarrow{x \in X_{\text{trg}}} \begin{pmatrix} x \\ 0 \\ x \end{pmatrix},$$

where 0 denotes the zero vector with dimensions equal to the dimensions of x. By augmenting the features like above, the domain adaptation task is left to the used model. For example, a linear model would give weights which have components in the shared source and target features space as well as in the feature spaces specialized for the source and target domains. This so-called frustratingly easy domain adaptation method can be straightforwardly generalized to the multi-source setting [11]. In it, the feature vectors are augmented as in

$$x \xrightarrow{x \in X_s} \begin{pmatrix} x \\ 0_1 \\ \vdots \\ 0_{s-1} \\ x \\ 0_{s+1} \\ \vdots \\ 0_S \\ 0 \end{pmatrix},$$
$$x \xrightarrow{x \in X_{\text{trg}}} \begin{pmatrix} x \\ 0_1 \\ \vdots \\ 0_S \\ x \end{pmatrix},$$

where 0_i refers to the zero vector of the dimension of x for the ith domain. In this representation, as before, the leading dimensions of the augmented feature vector refer to the shared feature space while the remaining dimensions are related to the domain specific features.

4.4.3 Kernel mean matching

Kernel Mean Matching (KMM) is an instance-based *unsupervised domain adaptation* method [26, 20] which has been extended to both to the supervised [39] and multi-source settings [59]. Unsupervised means that only unlabeled target data is available while instance-based refers to methods relying on the following relation. In machine learning, usually, models aim to minimize a loss which in most cases can be written as

$$L \equiv E_{(X,Y)\sim P_{\text{trg}}}[\mathcal{L}(Y, f(X))]\,,$$

where P_{trg} is the target distribution of the predictors X and predicted variable Y. Moreover, $\mathcal{L}$ is a loss function and f is a decision rule assigning a label to a predictor. If only unlabeled target data is available then in the spirit of importance sampling, we find that

$$\begin{aligned} L &= E_{(X,Y)\sim P_{\text{src}}}\left[\frac{p_{\text{trg}}(X,Y)}{p_{\text{src}}(X,Y)}\mathcal{L}(Y, f(X))\right] \\ &\approx \frac{1}{N_{\text{src}}}\sum_{i=1}^{N_{\text{src}}}\beta_i\mathcal{L}(y_i^{\text{src}}, f(x_i^{\text{src}}))\,, \end{aligned}$$

where P_{src} refers to the source distribution and $\beta_i = p_{\text{trg}}(x_i^{\text{src}})/p_{\text{src}}(x_i^{\text{src}})$ in the case that only the marginal distributions differ. Here, we assumed that the support of P_{trg} is contained in the support of P_{src}. In this way, the domain adaptation problem has been reduced to a problem of assigning instance weights to the training source examples. Instead of requiring knowledge of the data distributions, kernel mean matching solves this weighting problem by requiring that the empirical means of the source and target marginal distributions agree in the *Reproducing Kernel Hilbert Space* (RKHS) [42, 23]. That is, under soon to be specified constraints, the weights minimize

$$\left\| \frac{1}{N_{\text{src}}}\sum_{i=1}^{N_{\text{src}}}\beta_i\Phi(x_i^{\text{src}}) - \frac{1}{N_{\text{trg}}}\sum_{i=1}^{N_{\text{trg}}}\Phi(x_i^{\text{trg}}) \right\|^2$$

where Φ is a mapping to RKHS. In practice, this minimization task can be formulated as the quadratic programming model

$$\min_{\beta}\left[\frac{1}{N_{\text{src}}^2}\sum_{i=1}^{N_{\text{src}}}\sum_{j=1}^{N_{\text{src}}}\beta_i\beta_j k(x_i^{\text{src}}, x_j^{\text{src}}) - \frac{2}{N_{\text{src}}N_{\text{trg}}}\sum_{i=1}^{N_{\text{src}}}\sum_{j=1}^{N_{\text{trg}}}\beta_i k(x_i^{\text{src}}, x_j^{\text{trg}})\right] \tag{4.1a}$$

such that

$$0 \leq \beta_i \leq B\,, \forall i \in [1, N_{\text{src}}] \cap \mathbb{Z}\,, \tag{4.1b}$$

$$\left| \frac{1}{N_{\text{src}}} \sum_{i=1}^{N_{\text{src}}} \beta_i - 1 \right| < \delta\,, \tag{4.1c}$$

where k is the kernel function, and B and δ are hyperparameters to be chosen, for example, via cross-validation.

One problem with kernel mean matching is that it requires the construction of a $N_{\text{src}} \times N_{\text{src}}$ matrix and that the algorithm scales as $\mathcal{O}(N_{\text{src}}^3)$ [6]. These facts limit the applicability of the standard kernel mean matching to relatively small datasets. However, in [6], the authors have developed a sampling based approach to extended the regime in which kernel mean matching can be used in practice. They suggest the following algorithm

Data: covariates $X_{\text{src}}, X_{\text{trg}}$ and parameters m, η
Result: β
$s \leftarrow \frac{\ln \eta}{m \ln(1 - 1/N_{\text{src}})}$;
for $i \leftarrow 1$ **to** s **do**
  $X_{\text{src}}^{(i)} \leftarrow \text{sample}(X_{\text{src}}, m)$;
  $\beta^{(i)} \leftarrow \text{KMM}(X_{\text{src}}^{(i)}, X_{\text{trg}})$;
  $\beta \leftarrow \text{aggregate}(\beta^{(i)})$;
end
return normalize(β)

where sample performs repeated sampling of m instances out of X_{src}, KMM solves the minimization problem of Equations (4.1), aggregate does aggregation of the obtained weights and normalize does the final mean normalization depending on the aggregated weights.

While reducing the computational complexity of the kernel mean matching is important for this work, so is extending the method to handle multiple sources. In [59], the author suggests replacing the single-source kernel mean matching problem of Equations (4.1) with the minimization of

$$\left\| \sum_{s=1}^{S} \frac{1}{N_s} \sum_{i=1}^{N_s} \beta_i^s \Phi(x_i^s) - \frac{1}{N_{\text{trg}}} \sum_{i=1}^{N_{\text{trg}}} \Phi(x_i^{\text{trg}}) \right\|^2$$

such that

$$\min_{\beta} \left[\sum_{s=1}^{S} \sum_{s'=1}^{S} \frac{1}{N_s N_{s'}} \sum_{i=1}^{N_s} \sum_{j=1}^{N_{s'}} \beta_i^s \beta_j^{s'} k(x_i^s, x_j^{s'}) - 2 \sum_{s=1}^{S} \frac{2}{N_s N_{\text{trg}}} \sum_{i=1}^{N_s} \sum_{j=1}^{N_{\text{trg}}} \beta_i^s k(x_i^s, x_j^{\text{trg}}) \right]$$

subject to

$$\left| \beta_i^s - \frac{1}{N_s} \sum_{j=1}^{N_s} \beta_j^s \right| \leq \eta \,, \forall s \in [1, S] \cap \mathbb{Z}, i \in [1, N_s] \cap \mathbb{Z} \,,$$
$$\sum_{s=1}^{S} \frac{1}{N_s} \sum_{i=1}^{N_s} \beta_i^s = 1 \,,$$
$$\beta_i^s \geq 0 \,, \forall s \in [1, S] \cap \mathbb{Z}, i \in [1, N_s] \cap \mathbb{Z}$$

where β_i^s is the ith instance-weight and N_s is the number of instances in the domain s. Here, the hyperparameter η bounds the difference between individual instance weights and their mean for a given source. This comprises the *Multi-Source Kernel Mean Matching Method* (MSKMM).

4.5 Evaluation metrics

In this work, we have chosen to use two metrics to measure the discrimination ability of the considered models. These metrics are the Area Under the Receiver Operating Characteristic Curve (AUROC) and the Area Under the Precision-Recall Curve (AUPRC). In order to outline how these metrics can be calculated, we start by introducing the notion of a *confusion matrix*. The elements of a confusion matrix, see Table 4.1, are obtained from a probabilistic binary classifier by choosing a *probability threshold* θ above which a predicted probability reduces to the positive class and below which it results into the negative class. In terms of the elements of a confusion matrix, it is useful to define the ratios

		Actual	
		positive	negative
Predicted	positive	TP	FP
	negative	FN	TN
	total	P	N

Table 4.1: A confusion matrix has the predicted labels as rows and actual labels as columns. The True Positive (TP), False Positive (FP), False Negative (FN) and True Negative (TN) counts are its elements. Positive (P) and Negative (N) counts are summarized underneath the matrix.

$$\begin{aligned} \text{FPR} &= \frac{\text{FP}}{\text{FP} + \text{TN}}\,, \\ \text{TPR} &= \frac{\text{TP}}{\text{TP} + \text{FN}} \\ &= \text{recall}\,, \end{aligned}$$

which refer to the False Positive Rate and True Positive Rate, respectively. For a sequence of thresholds, the associated (FPR, TPR) pairs define a curve known as the Receiver Operating Characteric (ROC) curve. It can be shown that this curve has the properties that for a perfect classifier the curve is a step function at the origin while for a random classifier the curve is a line from the origin to the point (1,1) [15]. In this work, we will not look at these curves but rather the areas under them. Clearly, for a perfect classifier, this area, i.e., AUROC, is unity while for a random classifier it is one-half. In practice, AUROCs lie usually between these extremes, and the bigger the value, the closer the classifier is to the perfect classifier. It should be noted that AUROC is a only a measure of the ranking ability of the classfier. In fact, it measures the probability that a randomly chosen instance of the positive class is scored higher than a randomly chosen instance of the negative class [15].

Unfortunately, AUROC does not necessarily give a good measure of the discrimination of a classifier on an *imbalanced dataset* [12, 52]. Therefore, we will also introduce the AUPRC as our second metric. It measures the discrimination of the minority class classification only and is defined in terms of the precision

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}\,,$$

and recall. The precision-recall curve is obtained for a sequence of thresholds from the set of (recall, precision) pairs. This curve has the property that it is in practice like the step function of 1-recall for a perfect classifier while it is a constant, namely the prevalence $P/(P+N)$ of the positive class, for a random classifier [52]. Then, the area under it, i.e., AUPRC is a number ranging from the prevalence of the positive class to unity. Similarly to AUROC, it is a score, which means that the closer AUPRC is to unity the more discriminative the classifier. Finally, a couple of additional remarks are in order. Firstly, we note that unlike AUROC, AUPRC is sensitive to the prevalence of the positive class. This should be kept in mind when comparing AUPRCs. Secondly, optimizing an AUROC does not guarantee that the AUPRC is optimal as well [12].

5. Experiments

Computational experiments are needed to address the research questions of most machine learning -related studies. Here, our goal is to specify what are these experiments and how they are carried out. In particular, we will introduce three computational experiments which address the generalization and/or knowledge transfer aspects of our research questions. However, in order to have this discussion, we first need to discuss the ways we prepare the data. In addition, we will introduce the practical ways in which the machine learning models are realized in this work. This means that we will state the Python packages used and list the parameters of the models of this work.

5.1 Preprocessing

Preprocessing raw data is one of the first steps in building a machine learning model. Here, preprocessing consists of several subtasks like cohort selection, labeling, feature extraction and missing value imputation. In the following, we will go through these tasks in detail.

5.1.1 Cohort selection

Following [29], and due to our experimental setups requiring a hospital-wise split into sufficiently large training and testing set, we choose to examine hospitals containing at least 500 patient unit stays. See Figure 5.1 for a histogram of patient unit stays per hospital. In addition, we only consider patients of age at least 18 but do not impose a similar upper limit. This choice was motivated by the fact that the distributions of adolescent vitals as well as factors relating to mortality differ considerably from the corresponding adult or elderly characteristics. Other factors affecting the cohort are listed in the following subsection.

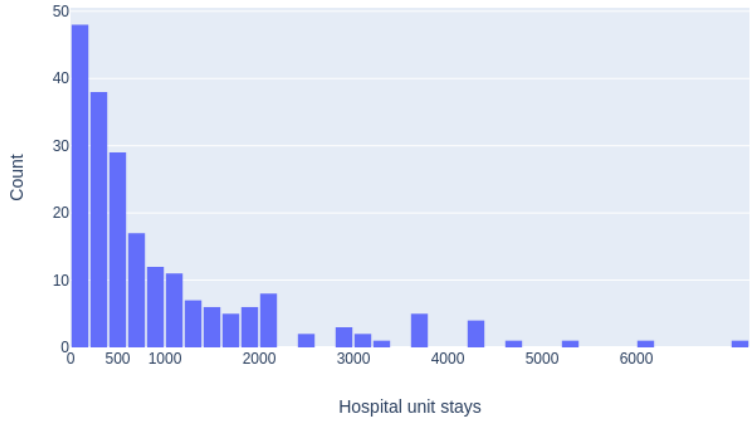

Figure 5.1: Histogram of hospital-wise grouped patient unit stays is shown. Only hospitals with at least 500 patient unit stays are considered in the modeling.

5.1.2 Labeling

In this work, we are concerned with the task of in-hospital mortality prediction. Therefore, in order to provide labels, we made the decision to define the time of death for the expired patients to be three hours before the unit discharge time. Expired patients refers here to patient units stays having `unitDischargeStatus` set to `Expired` in the `patient` table. The motivation behind this was that in some cases the vitals showed that the patient had already expired at the unit discharge. The positive labels were then assigned to expired patients one to seven hours before the above defined time of death. Samples at the time of death were dropped and not considered in the modeling. This was motivated by the fact that algorithmic mortality prediction is not needed so close to the time of death. We note that it would also have been possible to only implement the gap at test time but this was not experimented with.

5.1.3 Feature extraction

In this work, the primary features are the measured vital signs stored in tables `vitalPeriodic`, `vitalAperiodic` and `nurseCharting`. From `vitalPeriodic`, we have extracted the attributes `temperature`, `heartrate`, `respiration`, `systemicsystolic` and `systemicdiastolic`. We have appended `noninvasivesystolic` and `noninvasivediastolic` attributes from `vitalAperiodic` table to the attributes `systemicsystolic` and `systemicdiastolic`, respectively.

For each patient unit stay exceeding ten hours, we have calculated features with a one or five hour time window, depending on the used model, see Section 5.2, at one hour intervals measured from the unit discharge time. See Figure 5.2 for an illustration of the used setup with five hour aggregation windows. For each window, we have formed aggregate features by calculating the mean, standard deviation, minimum, maximum, first, last and trend value of each measurement. The trend has been calculated by fitting a simple linear regression model to the data inside each window and by extracting the corresponding slope. These aggregate features are our primary features.

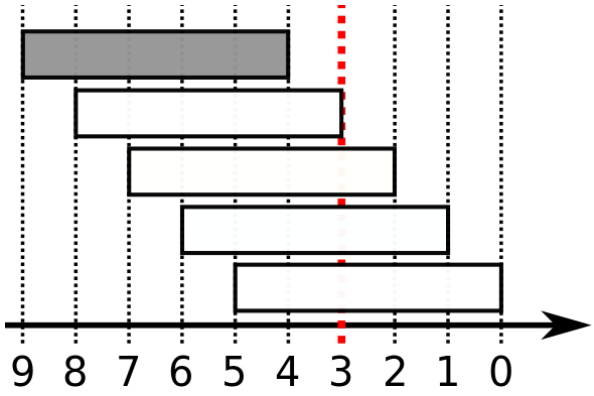

Figure 5.2: Illustration of the labeling and feature extraction process. Five hour aggregate features are calculated at one hour intervals counting from the unit discharge time. For expired patients, examples one to seven hours before the time of death (red line) are labeled positive (shaded rectangle) while the examples chronologically after this are dropped (rectangles).

Secondary features have been obtained from the `nurseCharting` table. We have chosen to use `nursingchartcelltypevallabel` attribute with values `Temperature` and `Glasgow coma score` to obtain the corresponding measurements. The actual values are found behind the `nursingchartvalue` attribute. Here, the `Temperature` values were found when `nursingchartcelltypevalname` took the value `Temperature (C)`. For `Glasgow coma score`, `nursingchartcelltypevalname` took values `Eyes`, `Verbal` and `Motor` which we took as the components of the Glasgow coma score. Similarly as before, we calculated aggregate features as described above and illustrated in Figure 5.2 with the exception that only the window mean was taken into account.

Prior to aggregating the features, we detected outliers and filtered them out. This was done by requiring that the acceptable values satisfied the requirements listed in Table 5.1.

In addition to the dynamic features described above, we took some static demographics into account. The attributes `gender`, `age` and `ethnicity` were extracted from the `patient` table. For `gender`, we combined the `Other` and `Unknown` values

	min	max
temperature	10	44
sao2	25	100
heartrate	20	250
respiration	0	100
systolic	10	300
diastolic	0	200
GCS (Eyes)	1	4
GCS (Verbal)	1	5
GCS (Motor)	1	6

Table 5.1: Feature values that lie between min and max are acceptable. Otherwise, the values are set to nan and imputed later.

and for `age`, we treated `> 89` as `90`. The final static features were obtained by dummy encoding the categorical variables.

5.1.4 Missing value imputation

Missing values are introduced in the design matrix during the aggregation process for the uniformly separated time points. We have chosen a simple two-way imputation process to fill the missing values for dynamic features. First, we fill for each patient unit stay the missing values with the previous value if it exists. Second, if no previous values exist, we impute each missing value of a feature with the median of that feature. We refer, e.g., to [17] for a discussion of more advanced methods for dealing with the irregularly and asynchronous sampled EHR datasets in which missing value imputation is not needed at all. In the case of categorical variables, for `gender`, we imputed with mode of that feature, and for `ethnicity`, we imputed with the value `Other/Unknown`.

5.2 Model specification

In this section, we will give the implementation details of the models introduced under Section 4.2. In particular, we will name the packages or frameworks used and discuss the model selection problem including the hyperparameter search.

5.2.1 RandomForestClassifier

In this work, we use the random forest implementation of the `scikit-learn` package known as `RandomForestClassifier` [45]. Unless otherwise stated, we use the default settings of the `RandomForestClassifier`. In order to build an optimal model, we do a hyperparameter search using cross-validation together with Bayesian optimization which is implemented in a `scikit-learn` compatible way in `BayesSearchCV` class of `scikit-optimize` package. Our hyperparameter search space is given in Table 5.2.

	values
n_estimators	$[300, 1000] \cap \mathbb{Z}$
max_depth	$[3, 10] \cap \mathbb{Z}$
min_samples_split	$[2, 10] \cap \mathbb{Z}$
min_samples_leaf	$[1, 10] \cap \mathbb{Z}$
max_samples	$[0.5, 0.99]$

Table 5.2: Hyperparameter search space for the `RandomForestClassifier`.

5.2.2 XGBClassifier

We chose `XGBClassifier` from the `xgboost` package as our implementation for the gradient boosted trees [7]. The `XGBClassifier` provides a `scikit-learn` friendly user interface and, unless otherwise stated, we use its default settings for our purposes. Our hyperparameter search space is given in Table 5.3. As with `RandomForestClassifier`, we use Bayesian optimization based search together with cross-validation estimation for finding the optimal hyperparameters.

5.2.3 LSTM

The third model used in this work is the Long Short-Term Memory (LSTM) network, see. 4.2.3. Specifically, we have chosen to treat our classification problem with a sequence-to-sequence LSTM model. In this case, each time-series of predictors $(x_n^t)_{t=1}^{T_n}$ is a sequence of length T_n and associated with an equal-length sequence of binary labels $(y_n^t)_{t=1}^{T_n}$. The binary label y_n^t indicates whether or not the patient stay ends in in-hospital death within one to seven hours from the hourly time label t. The predictors x_n^t have been obtained with hourly time windows as described in Section 5.1.3.

	values
n_estimators	$[300, 1000] \cap \mathbb{Z}$
learning_rate	$[0.001, 1]$
max_depth	$[3, 10] \cap \mathbb{Z}$
min_child_weight	$[1, 10] \cap \mathbb{Z}$
gamma	$[0, 1]$
colsample_bytree	$[0.1, 0.9]$
subsample	$[0.5, 0.99]$

Table 5.3: Hyperparameter search space for `XGBClassifier`. Note that the parameter learning_rate is sampled initially from the log-uniform distribution.

Our LSTM has a variable architecture chosen during model selection. In particular, we choose the number of stacked layers n_l of the model as well as the number of hidden units in each layer n_h using Bayesian optimization together with cross-validation. Note that each layer of the model has the same number of hidden units. See Table 5.4 for chosen values of n_l and n_h as well as the rest of the hyperparameter space.

In this work, we use the weighted binary cross-entropy loss

$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^{N} \Big(- p y_n^T \log \sigma_n - (1 - y_n)^T \log(1 - \sigma_n) \Big), \tag{5.1}$$

where y_n is the binary target vector, σ_n is the sigmoid output vector and $p = N_0 / N_1$ is the positive class weight. Here, N_1 and N_0 refer to the number of positive and negative instances, respectively. The positive class weight is introduced as a form of cost-sensitive learning to cope with the high class imbalance. It does so by imposing a higher cost for miss-classifying an instance of the positive class. The complete loss function is given by

$$\mathcal{L} + \lambda \|W\|^2 \tag{5.2}$$

where λ is a regularizing hyperparameter and W denotes the vectorized weights of the model. This l^2-regularization is introduced to prevent the model from overfitting. In this work, given the loss criterion, the model is subsequently trained using the Adaptive moment estimation (Adam) optimizer [33]. We have otherwise used the default values ($\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$) of `PyTorch`'s implementation of the Adam optimizer but treated the learning rate α of the method as a hyperparameter.

As noted above, we have chosen to use the `PyTorch` framework [44] to realize our LSTM model. In addition, we use the `Skorch` package to make the model

amenable for hyperparameter search with BayesSearchCV and to train the final models with the optimal hyperparameters. It should be noted that when fitting a model using `Skorch`, it automatically splits the given training set into an internal training and validation sets. This split gives by default a 80/20 ratio between these two sets. Importantly, in our work, we have used early stopping which is based on monitoring the loss on the validation set. We used `Skorch`'s default values for the patience and a threshold, namely 5 and 10^{-4}, which dictate when to stop the training. If these criteria are not met early, the training is limited to a maximum of 100 epochs.

	values
α	$[10^{-5}, 0.01]$
n_l	$1, 2$
n_h	$16, 32, 64, 128$
λ	$[10^{-5}, 0.01]$

Table 5.4: Hyperparameter search space for LSTM model: learning rate α, number of layers n_l, number of hidden units n_h and l^2-regularization λ. Note that both learning rate and regularization have been initially sampled from a log-uniform distribution.

5.3 Experiment specification

In this section, we will describe the computational experiments which we have conducted in this thesis. The main purpose of these experiments is to investigate the generalizability of the machine learning models specified in Section 5.2.

Before conducting any experiments, we have split the full data set into three pieces: a training, validation and test set. These sets form a 60/20/20 partition

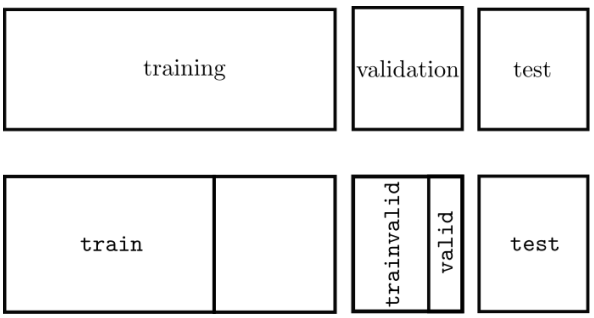

Figure 5.3: Illustration of the dataset splits. Top panel shows the 60/20/20 split into training, validation and test sets. Bottom panel illustrates a possible split into the `train`, `trainvalid`, `valid` and `test` sets.

of the full data set. First, the `test` set is sampled hospital-wise from the full data set attempting to preserve the hospital size distribution of the original data set. Hospital-wise means that all patient unit stays for a given sampled hospital belong to the `test` set. Second, the remaining data set is split patient-wise into the training and validation sets while trying to preserve the in-hospital mortality prevalence. Patient-wise means that all records corresponding to a patient unit stay belong to either of the split sets. We have realized both splits by relying on the stratified train_test_split function of scikit-learn [45].

I-II In these experiments, we go over data sets consisting of patient unit stays from n hospitals where n equals 1, 2, 3, 4, 5, 7, 9, 11 or 15. From the training set, we sample a data set consisting of patient unit stays from n hospitals. In experiment I, these sets form our true training sets. Instead, in experiment II, for each such set, we form the true training set by sampling m patient unit stays in a way which tries to preserve the in-hospital mortality prevalence. If m patient unit stays are not present in the sampled set of n hospitals, in the spirit of rejection sampling, we sample a new set of n hospitals until this condition is met. Sampling of each such obtained true training set, henceforth referred to as `train` set, from n hospitals is repeated twenty times to obtain point and uncertainty estimates for the evaluation metrics.

For each such sampled true training set, we train each of the machine learning models. The hyperparameters of such a model are obtained using Bayesian optimization based search with grouped k-fold cross-validation [24]. For each model, the hyperparameter search space is described in Section 5.2. The cross-validation groups are subsets of hospitals and there are a minimum of three

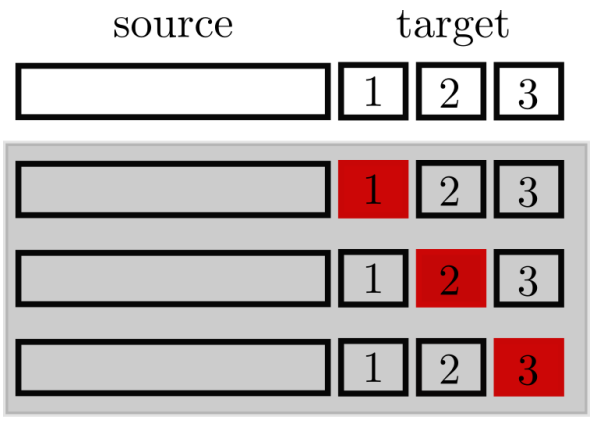

Figure 5.4: Illustration of the cross-validation scheme for the transfer learning experiment III. The top row shows the different parts/folds of the development set while the highlighted bottom rows illustrate the generated training and cross-validation sets. The enumerated cross-validation sets are further indicated with red color.

and the number n of hospitals folds. In the case that n equals one, we instead use stratified patient-wise cross-validation groups. Once trained, the models are evaluated on the training, validation and the `test` sets. Moreover, the evaluation on the validation set is split into two. We evaluate on the part of the validation set whose hospitals intersect with the hospitals of the training set as well as on the part which has different hospitals to the training set. In the figures, we refer to the former as the evaluation on the `trainvalid` set while the latter is referred to as the evaluation on the `valid` set, see Figure 5.3.

Sampling twenty training sets for n hospitals gives us the median and first/third quartiles of the AUROC and AUPRC, which are used as evaluation metrics. These quantities are compared against the number of hospitals to obtain insights into the generalizability of the machine learning models.

III In this experiment, we aim to investigate how transfer learning could be applied in the mortality prediction settings. In order to do so, both source and target hospitals need to be chosen. We have chosen the hospitals from the training set as the source hospitals and hospitals from the `test` set as the target hospitals. Setting up an evaluation procedure further requires a train/test split for the target hospitals. We have made a patientwise stratified 50/50 split of the `test` set into target training and test sets for each target hospital.

The experiment proceeds by going over source and target hospitals. In order to reduce the computational load, we have chosen the source part of the training set to be a randomly sampled set of five source hospitals containing randomly sampled hundred patient unit stays per source hospital. The target part of the training set consists of 10, 20, 40, 80, 160, 320, 480 or 640 patient unit stays from each target hospital. With such choice we go from the usual transfer learning regime to the situation in which source and target parts of the training set are of equal size. As with the prior experiments, the sampling of each such obtained true training set is repeated twenty times to obtain point and uncertainty estimates for the evaluation metrics.

Five different baseline and domain adaptation methods are investigated. The baseline methods labeled as 'source' and 'target' use only the source and target parts of the training set for training a machine learning model. The single-source domain adaptation baseline method 'weighted' uses uniformly weighted source instances for training the model, see Section 4.4.1. The two remaining models are labeled as 'fam' and 'kmm' corresponding to the feature augmentation method and multi-source kernel mean matching method, respectively. See Section 4.4.2 and 4.4.3 for their descriptions. It is further-

more noteworthy that the hyperparameters of the machine learning models have been obtained in the case of 'target', 'weighted' and 'fam' methods using the cross-validation procedure summarized in Figure 5.4. For 'source' and 'kmm', regular hospital-wise grouped 3-fold cross-validation is used.

As before, the twenty training sets for each source and target configuration gives us the median and first/third quartiles of the AUROC and AUPRC which are used as evaluation metrics. These quantities are plotted against the number of target hospital training patient unit stays to obtain an impression of the generalizability of the machine learning models.

6. Results

In the following, we first present the results of the experiments described thoroughly in Section 5.3 and then discuss the meaning and implications of these results as well as refer to existing results in the literature.

6.1 Experiment I

Experiment I contains a sequence of training and testing runs with increasing number of hospitals used for training. Briefly, we set the number of hospitals n, sample as many hospitals from the training set, train a model with the sampled hospitals data, and lastly test the model with four data sets. These data sets are the training set (`train`), a data set consisting of data for hospitals used in training (`trainvalid`), and two data sets consisting of data for hospitals not used in training (`valid`, `test`). The results of this experiment are shown in Figures 6.1 and 6.2.

In Figure 6.1, we show the area under the receiver operating characteristic curve (AUROC) for the `train`, `trainvalid`, `valid` and `test` sets plotted against the number of hospitals. In turn, Figure 6.2 displays the area under the precision-recall curve (AUPRC) for the same setup. Both median scores (lines) and interquartile ranges (IQR, areas) are shown for the three models: random forest (RF), gradient boosted trees (GBM) and long short-term memory network (LSTM). For the `train` set, the median AUROC varies between 0.74 and 1.0 depending on the model: GBM with the highest 0.9 to 0.97 and LSTM with the lowest 0.74 to 0.82 AUROC. For the same set, RF and GBM attain the highest AUPRCs within the range 0.25 to 0.61 while the AUPRC values for the LSTM are from 0.05 to 0.11. Comparing these values with the `trainvalid` scores: again GBM with the highest 0.81 to 0.84 and LSTM with the lowest 0.75 to 0.81 AUROC while RF and GBM have 0.07 to 0.13 and LSTM 0.04 to 0.08 AUPRC, we observe that GBM and RF show clear substantial overfitting while LSTM does not seem to suffer from it. For RF, this observation has preserved although we have tried both pre- and post-pruning with several hyperparameter search spaces for the Bayesian optimization to explore. However, we note that the `train` scores show that adding training data does decrease the

overfitting phenomenon. Returning back to `train` and `trainvalid` scores, we note that their high variation, cf., the `valid/test` set IQRs, seems to be coming from the sampling of the sets used for training and testing while for the `valid` and `test` sets the testing data is nearly fixed. Keeping this in mind, we observe that the `trainvalid` scores do not show significant increase when the number of hospitals is increased.

For the `valid` set, we observe that the median AUROC varies between 0.71 and 0.83 such that GBM has the highest range 0.77 to 0.83 while LSTM has the lowest range 0.71 to 0.80. At the same time, AUPRCs range from 0.03 to 0.10 with GBM again getting the highest 0.07 to 0.10 scores while LSTM scores the lowest with 0.03 to 0.07 AUPRCs. It is noteworthy that the prevalence is 0.007 for the union of the training and validation sets as well as for the test set. We see that the `trainvalid` scores dominate the `valid` scores but only slightly. This is in line with the assumption that a small training-validation set mismatch exists [43]. Another important observation is that in all models, up to some fluctuations in LSTM, the `valid` scores increase as the number of hospitals increase. Finally, to provide further assurance of the quality of results, we show the `test` set results. For the `test` set, AUROCs vary between 0.72 and 0.83 such that GBM has the highest 0.78 to 0.83 range and LSTM the lowest range from 0.72 to 0.80. AUPRCs are between 0.03 and 0.12: again GBM has the highest from 0.08 to 0.12 range while LSTM AUPRCs are between 0.03 and 0.09. These results are aligned with the results for the `valid` set which also should provide the correct testing scenario. It is however noteworthy that the `trainvalid` scores do not dominate the `test` scores which makes the notion of a mismatch questionable.

In summary, we observe that all methods clearly improve their performance on the test sets when more hospitals are added to the training set. Moreover, we consistently find that GBM outperforms RF which outperforms LSTM, and these observations hold for both AUROC and AUPRC. Severe overfitting is not observed for LSTM that is thus not the reason for its underperformance. Finally, we find that while GBM and RF seem to be relatively close to saturation, LSTM still benefits clearly from additional data for the larger datasets.

6.2 Hyperparameter search

Figure 6.3 shows the results of the hyperparameter search for Experiment I. Hyperparameter search has been carried out by Bayesian optimization, cf., Section 4.3. It is noteworthy that, with the exception of some LSTM parameters, the optimal hyperparameters are not consistent over the course of repeats of the experiment.

That is, different training sets lead to remarkably different configurations. We note that this may be due to the fact that the search has terminated prematurely as there were only 50 iterations allowed. Furthermore, we note that the results for LSTM have been obtained by a restricted hyperparameter search. We only did the search for the first of the twenty repeats of Experiment I and used the found optimal hyperparameters for the rest of the repeats. The reason for this was that by doing so the computational time remained feasible.

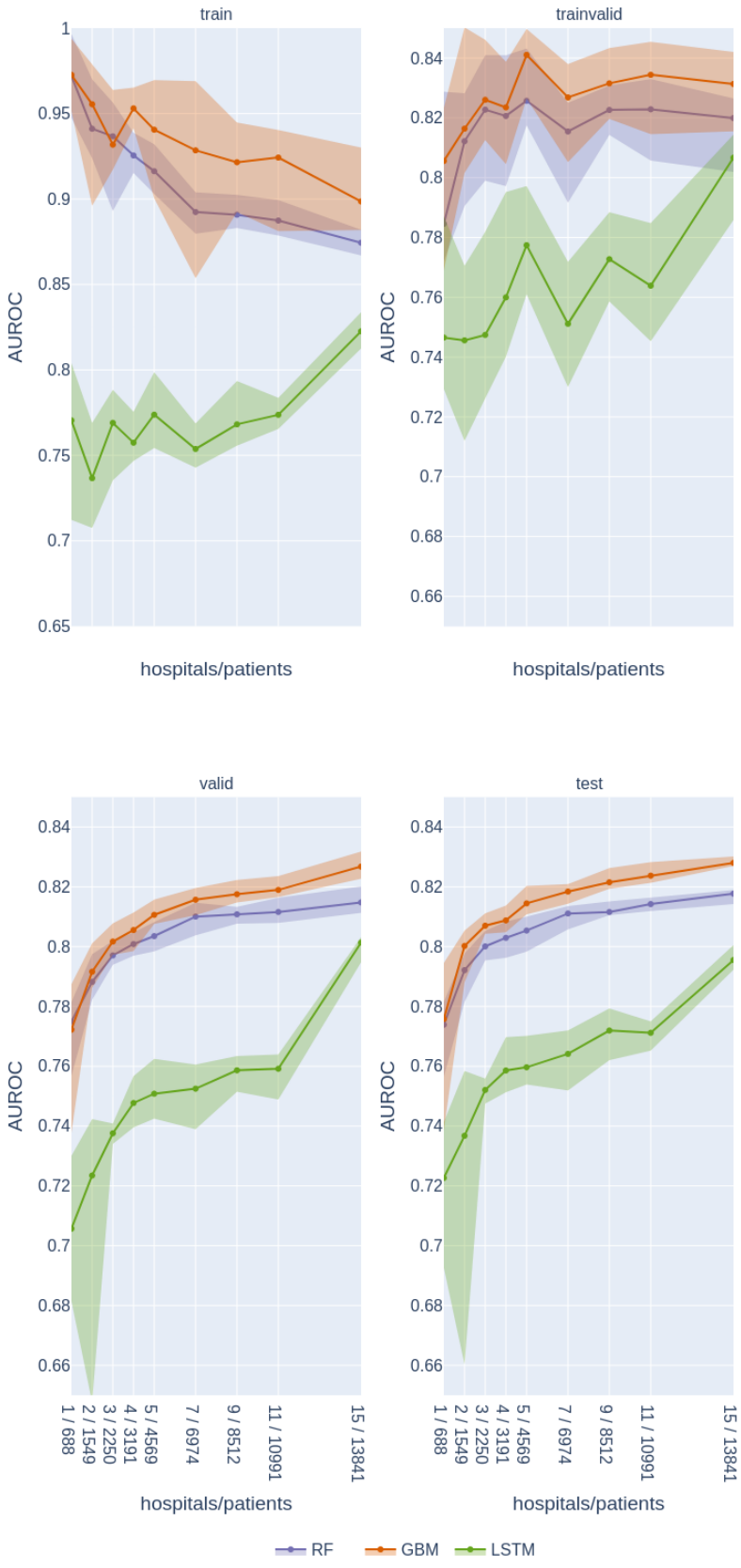

Figure 6.1: Experiment I: the area under the receiver operating characteristic curve (AUROC) shown as a function of the number of hospitals. The double x-axis shows the number of hospitals as well as the median number of patient unit stays. From left top to bottom right, `train`, `trainvalid`, `valid` and `test` scores, cf. 5.3, with median (line) and interquartile range (area).

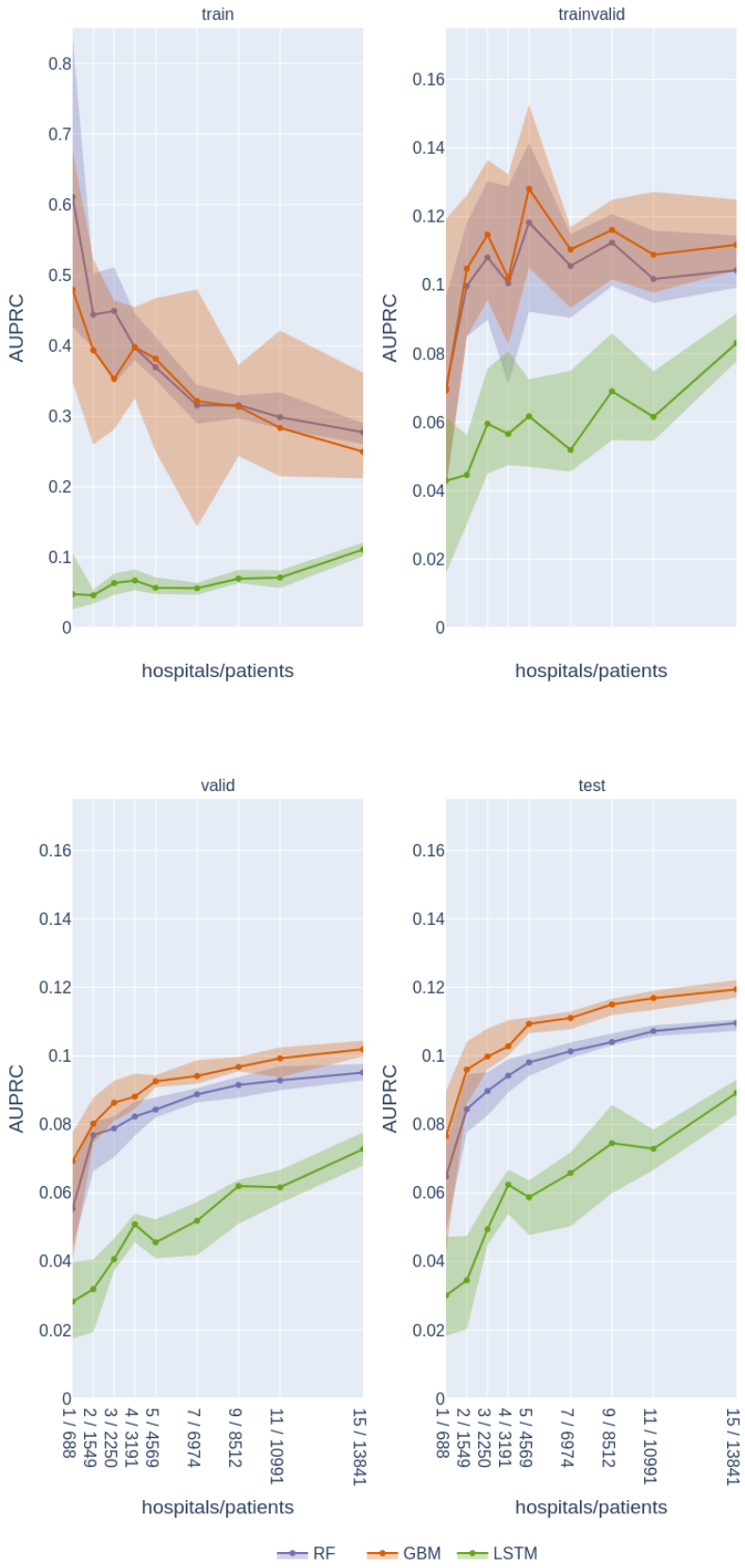

Figure 6.2: Experiment I: the area under the precision-recall curve (AUPRC) shown as a function of the number of hospitals. The double x-axis shows the number of hospitals as well as the median number of patient unit stays. From left top to bottom right, `train`, `trainvalid`, `valid` and `test` scores, cf. 5.3, with median (line) and interquartile range (area).

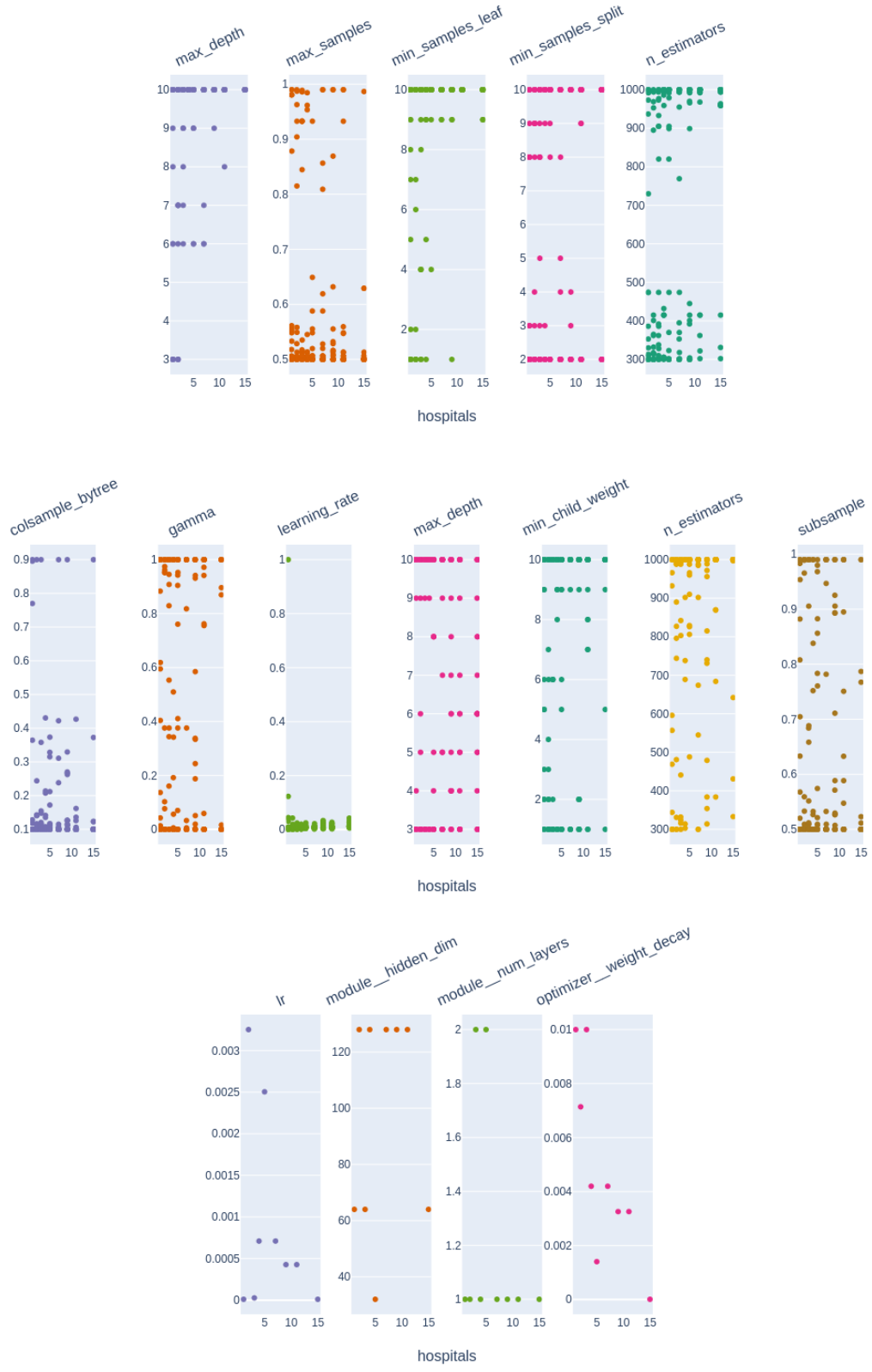

Figure 6.3: The distributions of the optimal hyperparameters in Experiment I. For RF and GBM, each point corresponds to a realized optimal hyperparameter and there are twenty samples for each hospital number. For LSTM, there is only a single optimal hyperparameter per hospital number. From top to bottom, random forest, gradient boosted trees and long short-term memory network.

6.3 Experiment II

In Experiment II, we have a setup which is almost identical to Experiment I. The only difference is that as we increase the number of hospitals n, we keep the number of patient unit stays in the `train` set at a constant value of 2000. As before, the model is tested on the training set (`train`), a data set consisting of data for hospitals used in training (`trainvalid`), and two data sets consisting of data for hospitals not used in training (`valid`, `test`). The results are shown in Figures 6.4 and 6.5.

Figure 6.4 contains the area under the receiver operating characteristic curve (AUROC) for the `train`, `trainvalid`, `valid` and `test` sets plotted against the number of hospitals. Figure. 6.2 displays the area under the precision-recall curve (AUPRC) for the same setup. Both median scores (lines) and interquartile ranges (IQR, areas) are shown for the two models: random forest (RF) and gradient boosted trees (GBM). We have decided to drop LSTM at this point because it did not show sufficient degree of performance in comparison to the tree-based models. Having said that, the `train` set median AUROC varies between 0.93 and 0.95 for RF while it is between 0.93 and 0.97 for GBM. Simultaneously, the median AUPRC lies between 0.42 and 0.49 for RF, and between 0.30 and 0.44 for GBM for the same set. In the `trainvalid` set, RF achieves decreasing median AUROC values from 0.82 to 0.80 and GBM shows values from 0.83 to 0.80. The `trainvalid` median AUPRCs for RF vary from 0.12 to 0.07 and for GBM from 0.11 to 0.08. Hence, here we observe the same overfitting as in Experiment I.

Proceeding to the `valid` set, RF attains a median AUROC range from 0.79 to 0.80 while GBM has values between 0.80 and 0.81. At the same time, the median AUPRC values range from 0.07 to 0.08 for RF and from 0.08 to 0.09 for GBM. Notice that both scores appear to be roughly constant with respect to increasing number of hospitals. In order to provide further evidence of the generalizability of the results, we also show the `test` set results. For the `test` set, the median AUROCs lie from 0.79 to 0.80 and from 0.80 to 0.81 for RF and GBM. The median AUPRCs range between 0.07 and 0.08 for RF and between 0.09 and 0.10 GBM for the same `test` set. These results are consistent with the `valid` set results.

In summary, we find that there is no significant increase in the metrics as more hospitals are added to the training set while keeping the number of patient unit stays constant.

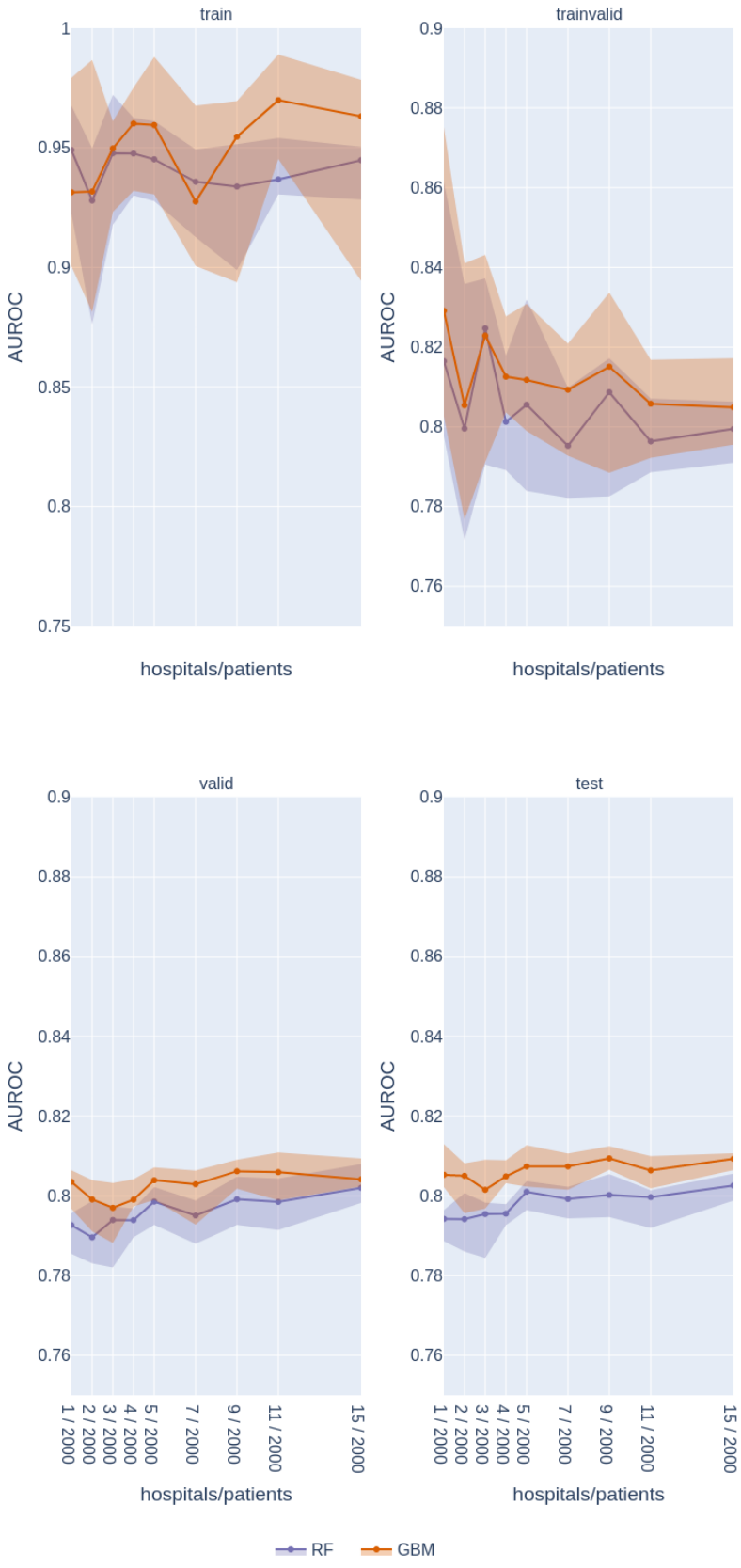

Figure 6.4: Experiment II: the area under the receiver operating characteristic curve (AUROC) shown as a function of the number of hospitals. The double x-axis shows the number of hospitals and the median number of patient unit stays. From top left to bottom right, `train`, `trainvalid`, `valid` and `test` scores, cf. 5.3, with median (line) and interquartile range (area).

Figure 6.5: Experiment II: the area under the precision-recall curve (AUPRC) shown as a function of the number of hospitals. The double x-axis shows the number of hospitals and the median number of patient unit stays. From top left to bottom right, `train`, `trainvalid`, `valid` and `test` scores, cf. 5.3, with median (line) and interquartile range (area).

6.4 Experiment III

Our last experiment is Experiment III which investigates the applicability of transfer learning to the task of in-hospital mortality prediction. This experiment focuses on a machine learning model's performance for only selected hospitals. We have chosen to focus on the RF model and on the four largest hospitals in the `test` set. The choice of RF is motivated by the fact that it contains less tunable hyperparameters than GBM which is valuable since some transfer learning methods even enlarge the hyperparameter space. In the experiment, a model is trained using the union of a part of the training set and a part of the `test` set. Our interest will be on the evaluation metrics when the size of the `test` set part of the training set is gradually increased. The metrics are calculated for the remaining test set not used for training. For further specifics of the setup see Section 5.3.

Having described our motivation and means, we point the reader to the results which are shown in Figures 6.6 and 6.7. Both median (lines) and interquartile range (IQR, area) are shown. In order to provide a baseline, we show results for the source and target methods. For the target method, the median AUROCs shown in Figure 6.6 vary between 0.66 and 0.82 while for the source method they lie in the range from 0.71 to 0.80. Similarly, the median AUPRCs shown in Figure 6.7 are between 0.01 and 0.11 for the target method while they range from 0.03 to 0.09 for the source method. Typically, the target method has lower scores for few patient unit stays but higher scores for many patient unit stays. This is the expected behavior. It varies at which values of patient unit stays the target scores overtake the constant source scores but for AUPRCs this typically happens somewhere between 160 and 320 patient unit stays. It is also particularly noteworthy that the between hospitals variation in both scores is large for all methods.

The weighted method provides an example of a single-source domain adaptation method applied to the mortality prediction task. For this method, the median AUROCs lie between 0.70 and 0.82 while the median AUPRCs range from 0.03 to 0.12. The trend is that the weighted scores dominate the target scores except for some AUROC cases. In comparison to the source method, the weighted method gives lower scores for few patient unit stays. We believe this is an artifact of the cross-validation procedure for the hyperparameter tuning as the method clearly has the potential to dominate both the target and source methods.

Then to the results of the two multi-source domain adaptation methods with the feature augmentation method (fam) first. For the AUROC, the median behavior shows that fam more often than not overperforms the target method. Often, it performs comparably to the weighted method. For few patient unit stays, the

source method has typically higher AUROCs but this we also believe is because of the cross-validation setup. For the AUPRC, the fam method dominates the target method, but compares more equally to the weighted method. Comparison to the source method is as in the case of the AUROC. Then, to the case of multi-source kernel mean matching (kmm) for which we have decided to not show the results. This is because our results were inconsistent, the method performed poorly, and they would only clutter the figures if shown. Similar observations about the quality of results were also made in the original source [59].

In summary, we observe that the source method is the best for very small amounts of target data. Moreover, we find that domain adaptation typically, see hospitals 73, 142 and 195 for AUROC, outperforms the source and target methods for intermediate amounts of target data. For large amounts of target data, the domain adaptation methods converge to the target method in terms of the evaluation metrics. Finally, we find no significant qualitative difference between the weighted and fam methods.

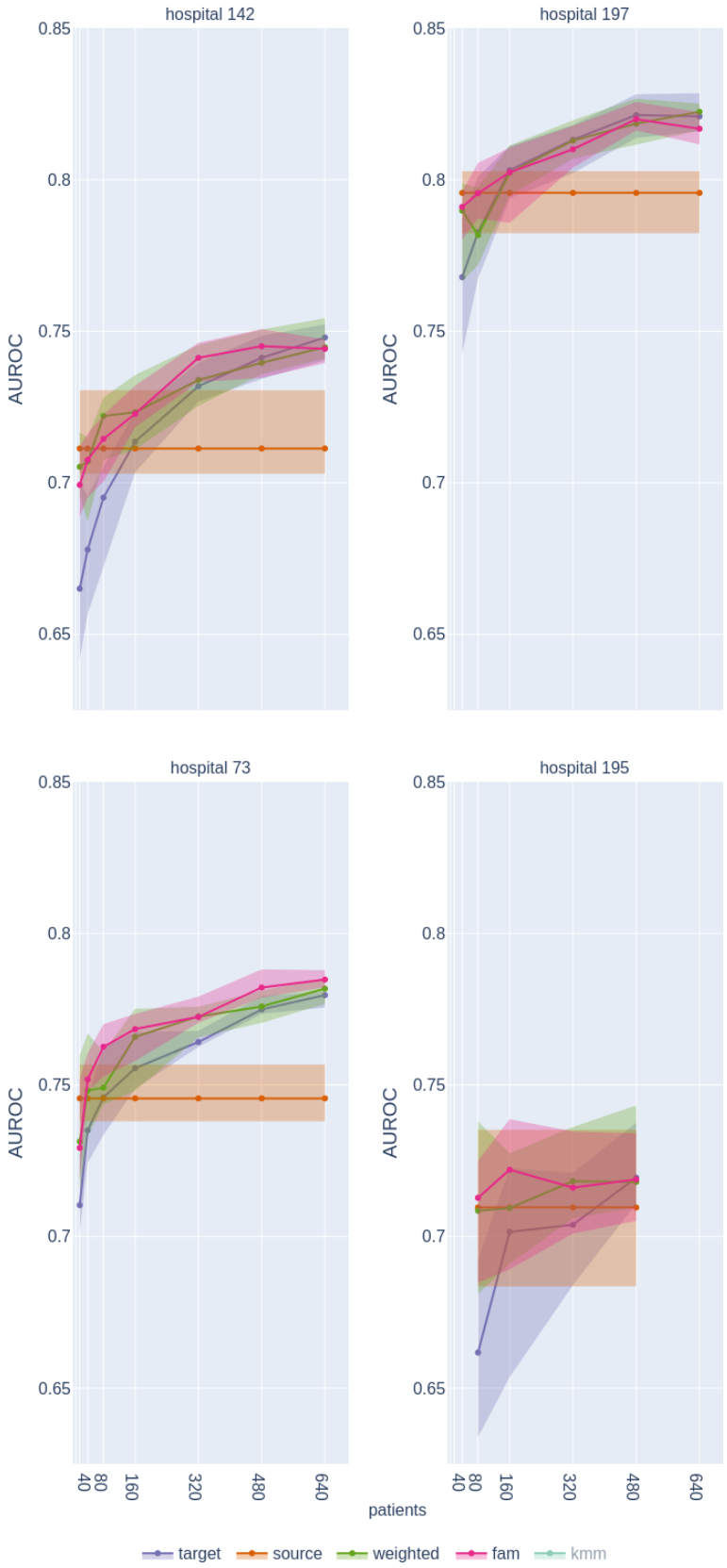

Figure 6.6: Experiment III: the area under the receiver operating characteristic curve (AUROC) shown as a function of the number of patient unit stays from the target hospitals. From top left to bottom right, four of the largest hospitals with median (line) and interquartile range (area).

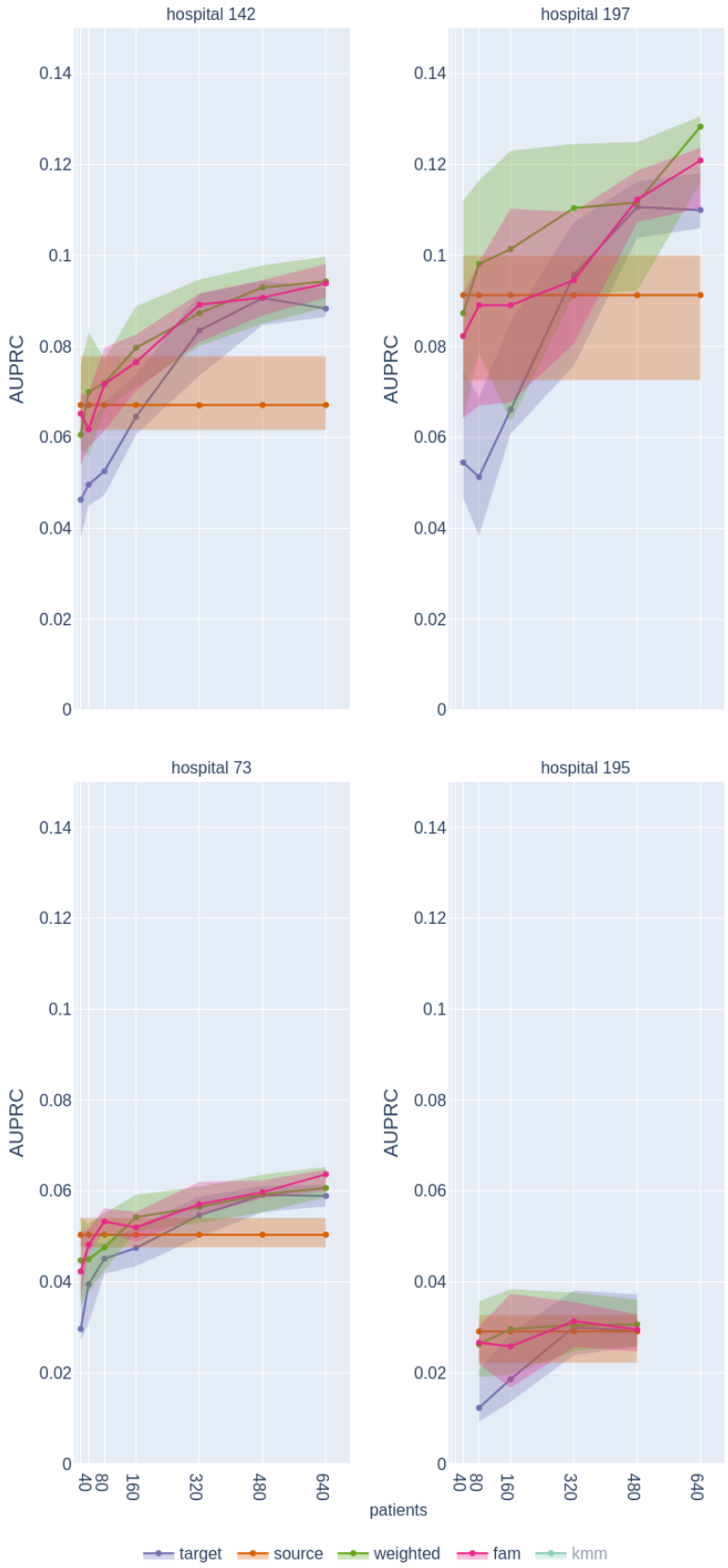

Figure 6.7: Experiment III: the area under the precision-recall curve (AUPRC) shown as a function of the number of patient unit stays from the target hospitals. From top left to bottom right, four of the largest hospitals with median (line) and interquartile range (area).

6.5 Discussion

In Experiment I, we investigated how well does an in-hospital mortality prediction model trained on data from some hospitals do on data collected from unseen hospitals. Our results showed that as the number of training hospitals is increased, the performance of the model as measured by AUROC and AUPRC increases as well. Gradient boosted trees achieved the best median AUROC of 0.83 and AUPRC of 0.12 on the test set with 15 hospitals used for training. This was at a test set prevalence of 0.007 and with stratified set splits to preserve the prevalence. In [29], Johnson *et al.* used a linear model and achieved a cross-validation AUROC of 0.854 and hospital-wise hold-out set AUROCs in the range from 0.688 to 0.933. Perhaps, the cross-validation score is more comparable to our results. We believe that one reason for it to be higher is that the authors took into account a considerably larger set of features. Another reason is their different approach for generating the samples. Finally, the cross-validation scores are expected to be higher than test scores. Cosgriff *et al.* studied calibration of severity of illness scores but also published the discrimination characteristics of several models [9]. Their AUROC and Average Precision (AP), which is comparable to AUPRC, were 0.907 and 0.596 for gradient boosted trees, respectively. Similarly as with [29], our assumption is that the difference in AUROC is due to more sophisticated features as well as differences in sample generation including labeling. For AP, we note that their prevalence is not mentioned, but we believe it is much higher since each patient unit stay is a sample in their scheme. This explains the considerable difference to our AUPRC.

One of the research questions of this work was to address the performance of deep learning models and compare it to the more traditional machine learning approaches. In our work, we found that LSTM networks do not perform as well as the tree-based ensemble models. This result is in a qualitative disagreement with the results presented in [37, 1] which highlight the success of recurrent neural networks. Moreover, our result also contradicts the results of [46]. In [46], the modeling setup is very similar to ours although the focus is on the emergency department and the predicted outcome is a cardiac arrest, transfer to ICU or death. The differences might be explained by the considerably smaller dataset which we have used. The fact that our LSTM results seem to benefit more from additional data when there are already several hospitals in the training set seem to support this explanation. Our hyperparameter tuning approach, which was chosen out of computational necessity, surely also limits the performance of our LSTM implementation. Moreover, our way of imputing missing values is not as sophisticated as the approach chosen in [46].

Our working hypothesis was that including more hospitals into the training set would improve the generalizability of the model. This was due to assumption that the distribution of the training set would cover more likely the distribution of the test set. In Experiment I, we noted the potential existence of a training and test set mismatch, i.e., difference in the distributions, for some range of hospitals used in training. In order to further test our hypothesis, we conducted Experiment II in which we kept the number of patient unit stays constant while increasing the number of hospitals used in training. Our results showed that both scores, AUROC and AUPRC, remained more or less constants as the number of hospitals was increased. This result is against our initial hypothesis. However, some criticism regarding the result is in order. Firstly, we believe that the experiment setup could have been refined. Instead of testing with all hospitals not in the `train` set, we could have tested with each hospital in the test set individually. This would have been more appropriate from the perspective of the training and testing set distributions. Secondly, we assume that by taking into account only the vitals, we have missed some of the important distributional differences between different policies in different hospitals.

In Experiment III, we studied the idea of utilizing target hospital data to improve the performance of the mortality prediction model on the target hospital. This study focused on transfer learning and, perhaps more accurately, on multi-source domain adaptation. Our results indicated that out of the tried baseline, and single- and multi-source domain adaptation methods, the weighted and feature augmentation methods were quite consistently the best. For example, for hospital 73, the median AUROC and AUPRC for the feature augmentation method increases nearly monotonously from 0.73 to 0.79 and from 0.04 to 0.06, respectively, as the number of patient unit stays was increased from 20 to 640. For comparison, in [29], the authors achieved an AUROC of 0.849 with the source(-only) method while the target(-only) method AUROC increased from 0.614 to 0.796 as patient unit stays were increased from less than 250 to 2400. Note in particular that our source results differ from these results because of a much smaller number of patient unit stays in our training set. Therefore, we cannot make similar conclusions, namely that source(-only) method outperforms the target(-only) method, as Johnson *et al.* do. Instead, our results are qualitatively in line with Curth *et al.* [10]. Moreover, although the used datasets were quite different from ours in [13, 10], we can still conclude that we achieved qualitatively similar results. That is, domain adaptation is beneficial and both weighted and feature augmentation method improve upon the baselines.

In addition to these results, we can draw some conclusions on the usefulness

of multi-source domain adaptation techniques, which is a research question that has not been addressed by the prior work. First of all, our single-domain (weighted) and multi-domain (feature augmentation, fam) domain adaptation methods yield very similar results. In addition, the domain adaptation results are only slightly better than the baselines. Together these observations imply that the domains, i.e., the distributions of the different hospitals, are perhaps only slightly different. This is in line with our observations about the questionable mismatch between the training and test sets in Experiment I. This can be potentially explained by the fact that we only consider the vitals which was already brought forth above. Beyond this, we would like to point out that our cross-validation setup was perhaps not ideal for the transfer learning methods. Instead of it, we should have maybe used the union of the source and target parts of the training set for obtaining the folds. Also, by using domain adaptation methods that take into account only differences in the marginal distributions, we most likely missed important differences due to care decisions. Instead, these would be included in the conditional distributions.

7. Conclusions

Severity of illness scoring systems are important from several aspects including but not limited to triage and resource allocation [36]. Machine learning has been used for data-based, real-time severity of illness scores [27, 28, 31]. However, it has been only until lately that multi-center datasets have become available [47] for systematic investigations of the generalizability of such models across hospitals. In this thesis, we have focused on studying the generalizability and knowledge transfer of real-time in-hospital mortality prediction models.

In order to do so, we have first reviewed prior work concerning mortality prediction, and generalizability, with traditional as well as deep learning models. We provide a discussion of the methods used in this work including the used models, model selection methodologies and transfer learning approaches. In addition, we give a detailed exposition regarding the data, i.e., eICU-CRD multi-center dataset, and its use for generating the results of this work.

The results section summarizes three computational experiments carried out in this thesis to address the generalizability and knowledge transfer of the used in-hospital mortality prediction models. Our main findings in these experiments are as follows. First, we find that increasing the number of hospitals in the training set increases the performance of the machine learning models for hospitals unseen by model. Out of the models investigated, gradient boosted trees perform the best. Second, when we increase the number of hospitals but keep the number of patient unit stays constant, we do not find a noticeable performance increase for the chosen evaluation setup. That is, increasing the heterogeneity of the training data does not seem to help. Third, despite that there is a negligible to small observed mismatch between training and test sets, we find that domain adaptation methods improve the performance of the models over simple baselines. Moreover, we found that a pooled single-domain adaptation method performs roughly equally to a multi-source domain adaptation method, which indicates that also domain differences range from negligible to small. It is noteworthy that this result is unique in the sense that we have not found other multi-source domain adaptation studies in the literature for this task.

We suggest that some problems related to the experiment setups should be fixed in the future to be more conclusive about the results. In particular, this concerns the evaluation in Experiments I and II which should be carried-out one hospital at a time. Our results also indicate that our models, especially the long short-term memory network, have the potential to benefit from more data than what we used. In Experiments III, the hyperparameter tuning setup should be changed as we point out in our discussion section. Moreover, a more extensive feature set should be utilized to provide a more realistic setup. This would also most likely make the distributional differences between the different hospitals larger and therefore increase the importance of multi-source domain adaptation .

From a practical perspective, given the results of this work, and if the regulations would allow, the transfer learning approach shows that such methods are a promising way for knowledge transfer between different hospitals. This would enable more accurate severity of illness scores to be deployed in circumstances where there is only scarcely data available.

Bibliography

[1] M. Aczon, D. Ledbetter, L. V. Ho, A. M. Gunny, A. Flynn, J. Williams, and R. C. Wetzel. Dynamic mortality risk predictions in pediatric critical care using recurrent neural networks. *CoRR*, abs/1701.06675, 2017.

[2] T. Alves, A. Laender, A. Veloso, and N. Ziviani. Dynamic prediction of ICU mortality risk using domain adaptation. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 1328–1336, 2018.

[3] S. Bailly, G. Meyfroidt, and J.-F. Timsit. What's new in ICU in 2050: big data and machine learning. *Intensive Care Medicine*, 44(9):1524–1527, Sep 2018.

[4] E. Brochu, V. Cora, and N. Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *CoRR*, abs/1012.2599, 12 2010.

[5] L. Celi, S. Galvin, G. Davidzon, J. Lee, D. Scott, and R. Mark. A database-driven decision support system: Customized mortality prediction. *Journal of personalized medicine*, 2:138–148, 09 2012.

[6] S. Chandra, A. Haque, L. Khan, and C. Aggarwal. Efficient sampling-based kernel mean matching. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 811–816, 2016.

[7] T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. Association for Computing Machinery.

[8] E. Choi, M. T. Bahadori, J. Sun, J. Kulas, A. Schuetz, and W. Stewart. RETAIN: An interpretable predictive model for healthcare using reverse time attention mechanism. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3504–3512. Curran Associates, Inc., 2016.

[9] C. V. Cosgriff, L. A. Celi, S. Ko, T. Sundaresan, M. Á. Armengol de la Hoz, A. R. Kaufman, D. J. Stone, O. Badawi, and R. O. Deliberato. Developing well-calibrated illness severity scores for decision support in the critically ill. *npj Digital Medicine*, 2(1):76, 2019.

[10] A. Curth, P. Thoral, W. van den Wildenberg, P. Bijlstra, D. de Bruin, P. Elbers, and M. Fornasa. Transferring clinical prediction models across hospitals and electronic health record systems. In P. Cellier and K. Driessens, editors, *Machine Learning and Knowledge Discovery in Databases - International Workshops of ECML PKDD 2019, Proceedings*, Communications in Computer and Information Science, pages 605–621. Springer, 1 2020.

[11] H. Daumé III. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[12] J. Davis and M. Goadrich. The relationship between precision-recall and ROC curves. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 233–240, New York, NY, USA, 2006. Association for Computing Machinery.

[13] T. Desautels, J. Calvert, J. Hoffman, Q. Mao, M. Jay, G. Fletcher, C. Barton, U. Chettipally, Y. Kerem, and R. Das. Using transfer learning for improved mortality prediction in a data-scarce hospital setting. *Biomedical informatics insights*, 9:1178222617712994–1178222617712994, Jun 2017. 28638239[pmid].

[14] F. Emmert-Streib, Z. Yang, H. Feng, S. Tripathi, and M. Dehmer. An introductory review of deep learning for prediction models with big data. *Frontiers in Artificial Intelligence*, 3:4, 2020.

[15] T. Fawcett. Introduction to ROC analysis. *Pattern Recognition Letters*, 27:861–874, 06 2006.

[16] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

[17] J. Futoma, S. Hariharan, and K. Heller. Learning to detect sepsis with a multitask gaussian process rnn classifier. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 1174–1182. JMLR.org, 2017.

[18] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[19] A. Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*. Studies in Computational Intelligence. Springer, Berlin, 2012.

[20] A. Gretton, A. Smola, J. Huang, M. Schmittfull, K. Borgwardt, B. SchÃ¶lkopf, J. Candela, M. Sugiyama, A. Schwaighofer, and N. Lawrence. Covariate shift by kernel mean matching. *Dataset Shift in Machine Learning, 131-160 (2009)*, 01 2009.

[21] P. Gupta, P. Malhotra, J. Narwariya, L. Vig, and G. Shroff. Transfer learning for clinical time series analysis using deep neural networks. *Journal of Healthcare Informatics Research*, 4(2):112–137, Jun 2020.

[22] A. Harriott and M. A. DeVita. *Virtual Mentor*, 16(12):969–975, 2014.

[23] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[24] T. Head, MechCoder, G. Louppe, I. Shcherbatyi, fcharras, Z. VinÃcius, cmmalone, C. SchrÃ¶der, nel215, N. Campos, T. Young, S. Cereda, T. Fan, rene rex, K. K. Shi, J. Schwabedal, carlosdanielcsantos, Hvass-Labs, M. Pak, SoManyUsernamesTaken, F. Callaway, L. EstÃšve, L. Besson, M. Cherti, K. Pfannschmidt, F. Linzberger, C. Cauet, A. Gut, A. Mueller, and A. Fabisch. scikit-optimize/scikit-optimize: v0.5.2, Mar. 2018.

[25] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, Nov. 1997.

[26] J. Huang, A. Gretton, K. Borgwardt, B. Schölkopf, and A. J. Smola. Correcting sample selection bias by unlabeled data. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 601–608. MIT Press, 2007.

[27] C. W. Hug and P. Szolovits. ICU acuity: real-time models versus daily models. *AMIA ... Annual Symposium proceedings. AMIA Symposium*, 2009:260–264, Nov 2009. 20351861[pmid].

[28] A. E. W. Johnson and R. G. Mark. Real-time mortality prediction in the intensive care unit. *AMIA ... Annual Symposium proceedings. AMIA Symposium*, 2017:994–1003, Apr 2018. 29854167[pmid].

[29] A. E. W. Johnson, T. J. Pollard, and T. Naumann. Generalizability of predictive models for intensive care unit patients. *CoRR*, abs/1812.02275, 2018.

[30] S. Kim, W. Kim, and R. W. Park. A comparison of intensive care unit mortality prediction models through the use of data mining techniques. *Healthcare informatics research*, 17(4):232–243, Dec 2011. 22259725[pmid].

[31] S. Y. Kim, S. Kim, J. Cho, Y. S. Kim, I. S. Sol, Y. Sung, I. Cho, M. Park, H. Jang, Y. H. Kim, K. W. Kim, and M. H. Sohn. A deep learning model for real-time mortality prediction in critically ill children. *Critical Care*, 23(1):279, Aug 2019.

[32] R. D. Kindle, O. Badawi, L. A. Celi, and S. Sturland. Intensive care unit telemedicine in the era of big data, artificial intelligence, and computer clinical decision support systems. *Critical Care Clinics*, 35(3):483–495, Jul 2019.

[33] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[34] A. A. Kramer, F. Sebat, and M. Lissauer. A review of early warning systems for prompt detection of patients at risk for clinical decline. *Journal of Trauma and Acute Care Surgery*, 87(1S), 2019.

[35] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[36] J. Lee, J. Dubin, and D. Maslove. *Mortality Prediction in the ICU*, pages 315–324. 09 2016.

[37] Z. C. Lipton, D. C. Kale, C. Elkan, and R. Wetzel. Learning to diagnose with LSTM recurrent neural networks, 2015.

[38] P. G. H. Metnitz, R. P. Moreno, E. Almeida, B. Jordan, P. Bauer, R. A. Campos, G. Iapichino, D. Edbrooke, M. Capuzzo, J.-R. Le Gall, and S. A. P. S. 3 Investigators. SAPS 3–from evaluation of the patient to evaluation of the intensive care unit. part 1: Objectives, methods and cohort description. *Intensive care medicine*, 31(10):1336–1344, Oct 2005. 16132893[pmid].

[39] Y.-Q. Miao, R. Araujo, and M. S. Kamel. Cross-domain facial expression recognition using supervised kernel mean matching. In *Proceedings of the 2012 11th International Conference on Machine Learning and Applications - Volume 02*, ICMLA '12, pages 326–332, USA, 2012. IEEE Computer Society.

[40] R. P. Moreno. Outcome prediction in intensive care: why we need to reinvent the wheel. *Current Opinion in Critical Care*, 14(5), 2008.

[41] R. P. Moreno, P. G. H. Metnitz, E. Almeida, B. Jordan, P. Bauer, R. A. Campos, G. Iapichino, D. Edbrooke, M. Capuzzo, J.-R. Le Gall, and S. A. P. S. 3 Investigators. SAPS 3–from evaluation of the patient to evaluation of the intensive care unit. part 2: Development of a prognostic model for hospital mortality at icu admission. *Intensive care medicine*, 31(10):1345–1355, Oct 2005. 16132892[pmid].

[42] K. P. Murphy. *Machine learning : a probabilistic perspective*. MIT Press, Cambridge, Mass. [u.a.], 2013.

[43] A. Ng. *Machine Learning Yearning*. 2018. https://www.deeplearning.ai/machine-learning-yearning/.

[44] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. In *NIPS-W*, 2017.

[45] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[46] T. Petäjä. Prediction of patient deterioration in the emergency department using recurrent neural networks. G2 pro gradu, 2019-03-11.

[47] T. J. Pollard, A. E. W. Johnson, J. D. Raffa, L. A. Celi, R. G. Mark, and O. Badawi. The eICU collaborative research database, a freely available multi-center database for critical care research. *Scientific Data*, 5(1):180178, 2018.

[48] S. Purushotham, W. Carvalho, T. Nilanon, and Y. Liu. Variational recurrent adversarial deep domain adaptation. In *ICLR*, 2017.

[49] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence. *Dataset Shift in Machine Learning*. The MIT Press, 2009.

[50] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[51] L. Rasmy, Y. Wu, N. Wang, X. Geng, W. J. Zheng, F. Wang, H. Wu, H. Xu, and D. Zhi. A study of generalizability of recurrent neural network-based predictive models for heart failure onset risk using a large and heterogeneous EHR data set. *Journal of biomedical informatics*, 84:11–16, Aug 2018. 29908902[pmid].

[52] T. Saito and M. Rehmsmeier. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, 10(3):e0118432–e0118432, Mar 2015. 25738806[pmid].

[53] J. I. Salluh and M. Soares. ICU severity of illness scores: APACHE, SAPS and MPM. *Current Opinion in Critical Care*, 20(5), 2014.

[54] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM J. Res. Dev.*, 3:210–229, 1959.

[55] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[56] J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'12, pages 2951–2959, Red Hook, NY, USA, 2012. Curran Associates Inc.

[57] K. Strand and H. Flaatten. Severity scoring in the ICU. *Acta anaesthesiologica Scandinavica*, 52:467–78, 05 2008.

[58] S. Sun, H. Shi, and Y. Wu. A survey of multi-source domain adaptation. *Inf. Fusion*, 24(C):84–92, July 2015.

[59] S. Takerkart. *A multi-source perspective on inter-subject learning. Contributions to neuroimaging.* PhD thesis, 2015.

[60] J.-L. Vincent and R. Moreno. Clinical review: scoring systems in the critically ill. *Critical care (London, England)*, 14(2):207–207, 2010. 20392287[pmid].

[61] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. A comprehensive survey on transfer learning. *ArXiv*, abs/1911.02685, 2019.

[62] J. Zimmerman, A. Kramer, D. Mcnair, and F. Malila. Acute physiology and chronic health evaluation (APACHE) IV: Hospital mortality assessment for today's critically ill patients. *Critical care medicine*, 34:1297–310, 06 2006.